UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE AUTOMAÇÃO E SISTEMAS

Fernando Gomes Papi

A Blockchain integration supporting the Transaction of Assets in Multi-Agent Systems

Florianópolis

2019

Fernando Gomes Papi

A Blockchain integration supporting the Transaction of Assets in Multi-Agent Systems

Dissertation presented to the Graduate Program in Automation and Systems Engineering in fulfillment of the requirements for the degree of Master in Automation and Systems Engineering.

Universidade Federal de Santa Catarina

Automation and Systems Department

Graduate Program in Automation and Systems Engineering

Supervisor: Prof. Jomi Fred Hübner, PhD.

Co-supervisor: Maiquel de Brito, PhD.

Florianópolis

2019

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Papi, Fernando Gomes
A Blockchain integration supporting the Transaction of Assets in Multi-Agent Systems / Fernando Gomes Papi ; orientador, Jomi Fred Hübner, coorientador, Maiquel de Brito, 2019.
87 p.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós-Graduação em Engenharia de Automação e Sistemas, Florianópolis, 2019.

Inclui referências.

1. Engenharia de Automação e Sistemas. 2. Sistemas Multi Agentes. 3. Blockchain. 4. Inteligência Artificial. 5. Transação de Ativos. I. Hübner, Jomi Fred. II. de Brito, Maiquel. III. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Engenharia de Automação e Sistemas. IV. Título.

Fernando Gomes Papi

A Blockchain integration supporting the Transaction of Assets in Multi-Agent Systems

The present work at Master's Degree level was evaluated and approved by an examining board composed of the following members:

We certify that this is the **original and final** version of this Dissertation which was deemed adequate to obtain the title of Master in Automation and Systems Engineering.

Werner Kraus Junior:53108523953

Assinado de forma digital por Werner Kraus Junior:53108523953
Dados: 2019.08.28 09:04:02 -03'00'

Prof. Werner Kraus Junior, PhD.
Graduate Program Coordinator

Jomi Fred [illegible]

Digitally signed by Jomi Fred Hubner:59633476968
Date: 2019.08.23 17:04:17 -03'00'

Prof. Jomi Fred Hübner, PhD.
Supervisor

Florianópolis, April 26th, 2019.

"I've developed a new open source P2P e-cash system called Bitcoin. It's completely decentralized, with no central server or trusted parties, because everything is based on crypto proof instead of trust...

...

The root problem with conventional currency is all the trust that's required to make it work. The central bank must be trusted not to debase the currency, but the history of fiat currencies is full of breaches of that trust. Banks must be trusted to hold our money and transfer it electronically, but they lend it out in waves of credit bubbles with barely a fraction in reserve. We have to trust them with our privacy, trust them not to let identity thieves drain our accounts. Their massive overhead costs make micropayments impossible.

...

With e-currency based on cryptographic proof, without the need to trust a third party middleman, money can be secure and transactions effortless.

...

The result is a distributed system with no single point of failure. Users hold the crypto keys to their own money and transact directly with each other, with the help of the P2P network to check for double-spending."

- Satoshi Nakamoto, 2008

"The one thing that's missing, but that will soon be developed, it's a reliable e-cash. A method where buying on the Internet you can transfer funds from A to B, without A knowing B or B knowing A. The way in which I can take a 20 dollar bill and hand it over to you and there's no record of where it came from. And you may get that without knowing who I am. That kind of thing will develop on the Internet."

- Milton Friedman, PhD., 1999

Resumo

Nossos dispositivos ficam mais inteligentes e conectados a cada dia. Conforme estes dispositivos começam a atuar de maneira autônoma em nosso mundo real, existe a necessidade de que estes dispositivos o façam de maneira benéfica, ao invés de maléfica. Por exemplo, agentes autonônos que atuam em mercados precisarão atuar conforme as regulações destes mercados. Há a necessidade de colocar estas regras em jogo para que estes agentes comportem-se como esperado. Necessita-se prover ferramentas para que os agentes envolvam-se em ações benéficas e, neste caso em particular, possam negociar seus recursos e ativos. Isto é, as interações entre agentes poderá envolver, além da troca de informação, a transação da propriedade de ativos.

Neste trabalho, desenvolveu-se um modelo no qual o conceito abstrato humano de ativos é inserido em Sistemas Multi-agente (MAS), tirando proveito da tecnologia Blockchain para modelor o mundo real na transação de ativos entre agentes autônomos. Para ilustrar como este modelo funciona, um problema é proposto e solucionado de forma incremental, realçando as vantagens trazidas pela integração do Blockchain. O modelo é capaz de regular o comportamento dos agentes, onde a confiança na transação do ativo não depende das crenças dos agentes envolvidos. Eles não precisam se preocupar com o valor a longo prazo dos ativos recebidos, enquanto a tecnologia Blockchain adiciona flexibilidade suficiente para que uma variedade de ativos possa ser negociada livremente.

Palavras-chave: Blockchain. Ativos. Regulação em Sistemas Multi-agentes.

Resumo Expandido

Introdução Conforme nossos dispositivos começam a interagir com o mundo físico, provendo informações e agindo autonomamente, surge a necessidade de que estes dispositivos o façam de maneira benéfica. Carros autônomos precisarão seguir nossos códigos de trânsito, agentes autônomos de investimentos necessitarão adequar-se às nossas regras de comércio e Veículos Aéreos Não Tripulados precisam garantir que a segurança das pessoas não seja colocada em perigo, e assim por diante. Portanto, é necessário que os agentes observem nossos códigos e leis para comportarem-se adequadamente. Agentes autônomos derivam suas ações de regras que maximizam um valor objetivo interno, e trabalharão incessantemente até que atinjam um determinado objetivo, o que pode causar problemas. Estes agentes necessitarão de recursos para cumprir seus objetivos, e dada a natureza finita de recursos, cria-se um cenário de "motivação de exploração" que pode levar estes agentes a tomar ações benéficas (como a troca de recursos) ou maléficas (como manipulação ou roubo de recursos). Para garantir que as ações tomadas pelos agentes sejam benéficas, precisa-se providenciar ferramentas para que estes agentes atuem amistosamente. No caso específico deste trabalho, espera-se que agentes interajam beneficamente na troca de recursos e ativos. Isto é, a interação entre agentes pode envolver, além da troca de informações, a troca da propriedade de ativos (aqui define-se "Ativo" como propriedade de uma pessoa ou empresa vista como tendo valor e que seja capaz de resolver dívidas ou compromissos).

A questão central deste trabalho é projetar Sistemas Multi Agentes (MAS) em que os agentes sejam capazes de atuar na troca livre de ativos. Em MAS, que por definição reunem mais de um agente, confiar em uma única entidade reguladora centralizada pode se mostrar impraticável. A hipótese considerada neste trabalho é que uma tecnologia chamada *Blockchain* pode oferecer o suporte necessário para a transação livre de ativos entre agentes sem a necessidade de tal entidade reguladora. Ainda que esta tecnologia seja proposta como parte da solução do problema, o conceito abstrato de ativos não é parte do Blockchain. O Blockchain é um registro seguro de transações, mas em sua essência, não passa de um "banco de dados" distribuído. Um registro no Blockchain não significa nada para um agente participante de um MAS a priori. Necessita-se de uma ferramenta que faça com que transações registradas no Blockchain condicionem o comportamento dos agentes para um resultado esperado. Não é a transação no Blockchain que torna o conceito de ativos concreto, e com isso em mente, necessita-se de uma ferramenta que torne tal conceito concreto para os agentes participantes do MAS.

Objetivos De maneira geral, este trabalho objetiva investigar e propor uma ligação entre Sistemas Multi Agentes e a tecnologia Blockchain para dar suporte a transação de ativos entre agentes autônomos. De maneira mais específica, o objetivo é aplicar o conceito de Instituições Artificiais para permitir o uso de MAS para a transação de ativos de qualquer natureza, potencializado pela tecnologia Blockchain, desenvolvendo-se um modelo onde o conceito abstrato de Ativos seja inserido em Sistemas Multi Agentes, tornando claro como o Blockchain modela o mundo real na transação de ativos entre agentes em um Sistema Multi Agentes aberto.

Este trabalho tentará responder as seguintes perguntas de pesquisa:

- **É possível representar a noção de ativos em MAS?**
- **A integração de um Blockchain com MAS é viável? Se sim, qual é o melhor papel ou modelo para um Blockchain em um MAS?**
- **É possível modelar e simular um problema ou sistema MAS + Blockchain e quais são as vantagens e desvantagens desta abordagem?**

Contribuições Este trabalho apresenta contribuições no desenvolvimento de um modelo de integração de duas tecnologias distintas e complementares: Sistemas Multi Agentes e Blockchain. Por ser uma tecnologia segura e confiável de computação distribuída, o Blockchain permite que agentes autônomos registrem transações de ativos, os quais neste trabalho tem a forma de *criptomoedas*, para atingir seus objetivos específicos. O modelo apresentado é flexível o suficiente para que qualquer categoria de ativos seja transacionada, desde que haja uma maneira de representá-la digitalmente. O trabalho ainda discorre sobre implicações, limitações e particularidades derivadas da aplicação da tecnologia Blockchain, como seus custos elevados de operação, baixa performance computacional se comparado a sistemas tradicionais de computação distribuída e limitação de escalabilidade. Ressalta-se, porém, através de exemplos ilustrativos, como um sistema de pagamentos entre agentes pode ser implementado em um MAS sem a dependência de entidades centralizadoras do mundo real (por exemplo, um banco). O trabalho é desenvolvido apresentando um cenário em que agentes precisam realizar tarefas específicas e receber um pagamento por estas tarefas. A forma de pagamento não está limitada ao escopo do cenário demonstrado, sendo que estes agentes podem utilizar o pagamento recebido pela tarefas em quaisquer outros sistemas que utilizem da mesma *criptomoeda*. O valor da criptomoeda e a persistência dos dados de pagamento são garantidos pelo uso da tecnologia de Blockchain escolhida para a implementação do cenário ilustrativo..

Conclusões De maneira geral, as conclusões deste trabalho são pautadas pelas respostas às perguntas de pesquisa previamente estabelecidas. O presente trabalho mostrou que é possível representar a noção de ativos em um MAS empregando o conceito de Instituição Artificial Situada (SAI). Também mostrou-se possível integrar a tecnologia Blockchain com MAS, analisando as vantagens e desvantagens desta abordagem. Concluiu-se que a melhor forma de fazer essa integração é através da utilização de artefatos de software específicos para integração entre MAS e Blockchain. Observa-se que a a utilização desta integração é conveniente apenas em operações em que as vantagens do Blockchain (segurança, confiabilidade, persistência de dados, disponibilidade) são criticas para a atuação dos agentes, dadas as suas desvantagens (custo, escalabilidade e performance). Por fim, mostrou-se possível modelar e simular um sistema MAS + Blockchain onde os agentes podem transacionar livremente ativos de forma confiável e segura.

Palavras-chave: Blockchain. Ativos. Regulação em Sistemas Multi-agentes.

Abstract

Our devices and gadgets get smarter and more connected to each other day by day. As these devices start acting autonomously in our world, there is a need for these devices to do so beneficially, instead of harmfully. For example, autonomous market agents will need to adequate themselves to our markets rules. Therefore, there will be a need to bring our codes and laws in play for these autonomous agents to behave as expected. These autonomous agents will need resources to achieve their goals, and since resources are generally scarce, this creates an “exploration drive" that will make them engage in either beneficial actions, like trading, or malicious actions, like theft. Since we must ensure that our artifacts behave well and abide to our rules, we need to provide tools for these agents to engage in beneficial actions, and as in this particular case, engage in beneficial trade of resources and assets. That is, the interactions between agents could involve, beyond the exchange of information, the transaction of assets’ property.

This work has developed a model where the human abstract concept of Asset is inserted in Multi-Agent Systems (MAS), taking advantage of the Blockchain technology to model the real world in a transaction of assets among agents in a MAS. To illustrate how this model works, an illustrative problem was proposed and solved incrementally to highlight the advantages brought by the integration of the MAS with the Blockchain. The model was capable of regulating the behaviour of agents, where the trustworthiness of the financial transaction does not depend on the belief of the agents involved. They do not need to be concerned with the long-term value of the assets they are receiving, and the Blockchain adds enough flexibility so that a variety of assets can be exchanged using this approach.

Key-words: Blockchain. Assets. Regulation in Multi-agent Systems.

List of Figures

Figure 1 – An example of an Artificial Institution 32
Figure 2 – Overview of the SAI Model 33
Figure 3 – Example of the SAI model - an Auction 33
Figure 4 – Schematic of a blockchain 34
Figure 5 – Schematic of a Bitcoin transaction, extracted from Evry's Blockchain Whitepaper 38
Figure 6 – Ethereum Project's Logo 40
Figure 7 – The historical price of Ethereum in American Dollars on 23/02/19. 42
Figure 8 – A table comparing Blockchain properties and MAS requirements, taken from [14] 44
Figure 9 – Model of the Blockchain as a generic environment 47
Figure 10 – Model of the Blockchain as specific artifacts in the environment 49
Figure 11 – The whole integration of the system 50
Figure 12 – Goals diagram for the *build-a-house* example 56
Figure 13 – Assignment of goals to Agent Roles 57
Figure 14 – Auction Artifact 57
Figure 15 – The outcome of the MAS, a digital house built for Giacomo 58
Figure 16 – Definition of *play(A,R,Gr)* status function and constitutive rules derived from it 60
Figure 17 – Definition of *commited(A,M,S)* and *achieved(S,G,A)* status functions and constitutive rules derived from them 61
Figure 18 – Bank Artifact 62
Figure 19 – Definition of constitutive rules that regulate payments in the MAS 63
Figure 20 – Excerpt of the Bank Smart Contract 65
Figure 21 – Bank Artifact Interfacing Ethereum's network 66
Figure 22 – QR Code for https://kovan.etherscan.io/address/0xfd9acf15dea 66

List of abbreviations and acronyms

IOT	Internet of Things
MAS	Multi-agent Systems
BTC	bitcoin (unit)
ETH	Ether (unit)
P2P	Peer-to-Peer
BDI	Beliefs, Desires, Intentions
ADICO	Attributes, Deontic Operator, Aim, Conditions, Or-Else
SAI	Situated Artificial Institutions
PoW	Proof of Work
JSON-RPC	JavaScript Object Notation - Remote Procedure Call
EVM	Ethereum Virtual Machine
USD	United States of America Dollars
JBBA	Journal of Brittish Blockchain Association
IEEE	Institute of Electrical and Electronics Engineers
UAV	Unmanned Aerial Vehicles
AI	Artificial Intelligence
CCP	Contract, Construct, Pay

List of symbols

X	Set of Environmental Elements
A_χ	Set of agents
E_χ	Set of events
S_χ	Set of states
Φ	Set of status-functions
E_ϕ	Set of events status-functions
A_ϕ	Set of agents status-functions
S_ϕ	Set of states status-functions
K	Set of Constitutive Rules
κ	Constitutive Rule
$O(\nu)$	Big O notation
τ	goal
US$	United States of America Dollars

Contents

1 INTRODUCTION **21**
1.1 Research Objectives **23**
1.2 Nomenclature **24**
1.3 Document Structure **24**

2 THEORETICAL BACKGROUND AND LITERATURE REVIEW **27**
2.1 Multi-Agent Systems (MAS) **27**
2.1.1 Environments 28
2.1.2 Coordination of agents and Organization 28
2.1.3 Open Multi-Agent Systems and Regulation 29
2.2 Artificial Institutions **30**
Environmental Elements 31
Status Function 31
Constitutive Rules 32
2.3 Bitcoin and fundamentals of Blockchains **32**
2.3.1 Basic concepts of Bitcoin’s Blockchain 33
Basic architecture of a blockchain 34
Transactions 35
Block Creation and Mining 36
Distributed Consensus 38
2.4 Smart Contracts **39**
2.4.1 Ethereum Virtual Machine and JSON-RPC Protocol 39
2.4.2 The “value" of Ether 41
2.5 Literature Review and Related Works **42**

3 CONCEPTUAL MODEL AND SYSTEM DESIGN **45**
3.1 Conceptual models for MAS and Blockchain **45**
3.1.1 Agents, Environment, Organization and Institution 45
3.2 Integrating the Blockchain to the system **46**
3.2.1 Blockchain as a generic environment 46
3.2.2 Blockchain instrumenting application artifacts 48
Interface Artifact and Smart Contract 50
3.3 Conclusion **53**

4 AN APPLICATION EXAMPLE **55**
4.1 Original problem with a simple payment extension **55**
4.2 SAI Regulation of the problem **59**
4.2.1 Introducing Payments 60
4.3 The Blockchain Integration **63**

4.3.1 The Smart Contract and The Interface Artifact 64
4.3.2 The flexibility of Smart Contracts 67
4.4 Conclusion 67

5 CONCLUSIONS 69
5.1 Future Work 71

BIBLIOGRAPHY 73

APPENDIX 77

APPENDIX A – CONSTITUTIVE RULES 79

APPENDIX B – SMART CONTRACT CODE 85

1 Introduction

The area of Artificial Intelligence in general, and the area of Multi-Agent Systems - MAS - in particular, aims for the development of autonomous entities - *agents* - capable of flexible and autonomous action in a dynamic context, usually a context containing other autonomous agents [28]. As our devices and gadgets get smarter and more connected to each other day by day, a new market emerges and with it new issues and opportunities arise at the same velocity. Gartner published a survey [30] estimating that this new market, now called “Internet of Things" - *IoT* will include over 25 billion connected devices and generate revenue exceeding US$300 billion by 2020. As these devices start engaging with the physical world, by providing information and acting autonomously, there is a need for these devices to do so beneficially, instead of harmfully. Self driving cars will need to follow our traffic codes, autonomous market agents will need to adequate themselves to our markets rules, autonomous delivery drones, if they ever see the light of day as a mass market, will need to make sure they do not harm anyone in the process, and so on. Therefore, there will be a need to bring our codes and laws in play for these autonomous agents to behave as expected.

In their most basic design, autonomous agents will derive their actions from what maximizes their internal objective value, whatever it is. If they are designed to achieve a particular goal, they will relentlessly try to do so, and it is clear why that might bring a wide range of problems. According to [35], these autonomous agents, being physical or not, will need resources to achieve their goals. Since resources are generally scarce, this creates an “exploration drive" that will make them engage in either beneficial actions, like trading, or malicious actions, like manipulation, theft, domination or murder. Since we must ensure that our artifacts behave well and abide to our rules, we need to provide tools for these agents to engage in beneficial actions, and as in this particular case, engage in beneficial trade of resources and assets. That is, the interactions between agents could involve, beyond the exchange of information, the transaction of assets’ property. But what defines an asset and what does property mean?

The definition of asset, according to Cambridge Dictionary [19], is “property owned by a person or company, regarded as having value and available to meet debts, commitments, or legacies". Therefore, an asset is an abstract concept that has no meaning *per se*, because value is subjective, according to some economists, such as [42]. Though the definition of asset might be quite straightforward, the concept of property can be somewhat trickier. Property implies that other parties agree that an asset belongs to something or someone. This possession of the asset is generally backed by property rights, where a central power or arbiter (generally, a government of some sort) makes that possession official, e.g., when official documents provided by a city attest that someone is the rightful owner of a house. Though not all property need to be made official by a central arbiter, the concept extends to small possessions as well. If someone steals a computer or cellphone from another person, the robber will still be held accountable for the theft of the asset in question.

The question that arises at this point is how can we design Multi-Agents Systems where agents are able to engage in a free trade of assets. When we are talking about a massive number of autonomous entities dealing and trading, relying on a single centralized entity to ensure that a specific asset is property of an agent could be unfeasible. The hypothesis is that a technology called **Blockchain** can offer the support for the transaction of assets between agents without relying in such centralized power.

After the general collapse of the global economy in 2008, and the subsequent bailout of large banks by governments, an anonymous person (or, more likely, a group of people), under the pseudonym of Satoshi Nakamoto, released a white-paper [31] describing the world's first fraud-proof digital currency: the Bitcoin. The Bitcoin is, as the title of that document states, a Peer-to-Peer Electronic Cash System. A decade later, the underlying technology that made Bitcoin viable is now called the **Blockchain**. This technology is essentially a data structure and computing protocol that contains the register of all of the currency's transactions, that is, every and each modification of the state of this network over time. Its cryptographic nature makes the blockchain an unhackable time-stamped database of transactions, disseminating trust along a decentralized network of nodes [43]. Such characteristics enable a great number of applications that require or would greatly benefit from decentralized consensus and integrity of registers or data.

Beyond currencies and banks, blockchains are speculated as a game-changing technology in domains such as Smart Grids, Supply Chain Management, Transfer of Medical Records, Lotteries, Virtual Auctions and Marketplaces, Virtual Collaborative Enterprises, Crowdfunding Platforms, Reputation Systems and so on [43, 32]. What all these domains have in common is that they have a distributed or decentralized nature. These are also common targets of applications of Multi-Agent Systems (MAS). These could be Open MAS, a category of MAS where agents are free to enter and leave and these agents are not known in design time [2]. In these systems, there is no guaranteed cooperation among these agents. Even the opposite could be true, there could be malicious agents that enter and leave the Open MAS.

Another problem that arises is that these abstract concepts of assets and property are not a part of Blockchain. The blockchain is a secure register of transactions, but at the very core, it means nothing to an agent participating in a MAS, from the point of view of this MAS. We need tools that make these transactions on blockchain condition agents' behaviour to a desired outcome. For example, even when an agent receives an incoming transaction on the blockchain for having performed some action, that transaction is not automatically related to some desired state of the MAS. It is not the transaction on the Blockchain that makes the concept of assets concrete, and with this in mind, we need a tool that makes such concept concrete to the MAS and the agents participating on it.

1.1 Research Objectives

From the introduction, we have discussed over two problems:

- Currently, Multi-Agent Systems have little to no frameworks that support transactions of assets between agents;
- Even when some specific technology is hypothesized to offer this support, it might not be sufficient from the point of view of the MAS, i.e., it might not be sufficient to support agents with their objectives and condition their behaviour, or even prevent harmful agents from operating.

In a short sentence, this work aims to investigate and propose a link between Multi-Agent Systems and the Blockchain technology to support the transaction of assets between autonomous agents. The general idea is that Multi-Agent Systems have something to gain from Blockchain in terms of security, transparency and trust among agents pursuing their goals. On the other hand, Blockchain has takeaways from Multi-Agent Systems regarding framework for development and theoretical base.

More specifically, this work aims to apply the concept of Artificial Institutions [8] to enable the usage of MAS for transaction of assets of any nature, supported by the Blockchain technology. The goal is to develop a model where the human abstract concept of Asset is inserted in Multi-Agent Systems (MAS), and to clarify how could the Blockchain model the real world in a transaction of assets among agents in an Open Multi-Agent System.

As stated before, the common development tools of Multi-Agent Systems lack some of the properties that could be provided by a blockchain. This could be a way to enforce cooperation, or at the very least, prevent malicious agents from harming the global objective of other agents. This work will try to answer the following research questions:

- **Is it possible to represent the notion of assets?** Since assets are not a concrete concept, can we represent this meaning in MAS and provide the necessary tools for agents to reason and act upon this concept?
- **Is integrating a blockchain into a MAS viable? If so, what is the best role or model for Blockchain in a MAS?** The blockchain provides many features that potentially indicate that it is a good fit for the transaction of assets in a MAS. There are different possibilities for integrating the blockchain, and this work will assume one of some possible implementations and evolve over this assumption.
- **Is it possible to model and simulate a problem or system with MAS + Blockchain and what are the advantages and disadvantages of this approach?** In order to

validate the adoption of a Blockchain by MAS, this work provides a practical implementation upon a problem that is often approached by MAS, extended by the integration with a blockchain.

To achieve the goals and answer the questions previously stated, a qualitative approach is pursued. First, a few possible models for the integration of the Blockchain are proposed and discussed. One of those models was selected and implemented. To test the validity of that model, a practical example is proposed, modeled and implemented.

The practical application consists of the construction of a Digital House by several different agents, a common example in MAS [8]. From the basic implementation of this example, only a layer of payments for the construction of the house is done, as a basic starting point. Then, those payments are regulated in the MAS, but everything still is done without using a blockchain. Finally, a layer of payments on the blockchain is added. Some conclusions are drawn from these experiments.

1.2 Nomenclature

For a better comprehension of this text, some clarifications about the nomenclature used are provided. Whenever possible, this work uses the most common and accepted words for the concepts presented. *Bitcoin* with uppercase *B* is used for the concept, the project and the network itself, the idea of this digital currency, whereas *bitcoin* (also, *BTC* and *btc*) generally refers to the token, the currency of the network. For example, "Using Bitcoin's network, Alice has just sent 2.0 bitcoins to Bob and 1.0 BTC to Giacomo.". Similarly, *Ethereum* with capital *E* refers to the overall concept, project and network. Differently from Bitcoin's case, Ethereum's token has a different name from the protocol. The name of its currency is *Ether*, generally contracted to *eth* or *ETH*. Bitcoin and Ethereum are Blockchain projects, bitcoin and Ether have prices.

Blockchain refers to the overall concept, technology and theory, and *blockchain* refers to the actual implementation of a distributed ledger over a P2P network. For example, "Alice is really interested in studying Blockchain theory, and is actually working in some improvements for the Ethereum blockchain.". In sentences where both terms could be used without loss of comprehension, the broader term *Blockchain* was preferred. The uppercase word was also used in Chapters' and Sections' titles. For simplicity, the words *btc*, *ether*, *eth* and *ethereum*, in lowercase, were avoided.

1.3 Document Structure

This section briefly provides what is discussed over the next chapters.

Chapter 2 provides the necessary Theoretical Base of Multi-Agent Systems, Regulation of MAS, Artificial Institutions and Blockchains and the connections between these systems.

Chapter 3 proposes a Conceptual Model for a Multi-Agent Systems with the integration of a Blockchain. This is done mainly taking into account advantages and disadvantages of the Blockchain technology.

Chapter 4 brings the proposition of an illustrative example. This example is modeled, implemented and tested, and some discussions and conclusions are drawn from the application.

Finally, in Chapter 5 this work is summarized and a Conclusion is drawn from the models developed, also providing some future works that dervie from this work.

2 Theoretical Background and Literature Review

The objective of this chapter is to provide a theoretical background on the concepts that ground this research. Multi-Agent Systems are briefly described (section 2.1), but Artificial Institutions and the Blockchain Technology are discussed to a greater extent in sections 2.2 and 2.3 respectively. In the end of the chapter, a review of literature and related works is presented.

2.1 Multi-Agent Systems (MAS)

A common definition of agent is a computer system, situated in some kind of environment, that is capable of autonomous actions in order to achieve some desired goal. Agents act upon and have perceptions of the ambient around them. They normally don't have full control of their surroundings, but rather a limited extension of influence. Since their environment could be constantly changing, agents need to be adaptable, to an extent, and autonomous: no human or other agent should be able to interfere with its objective of fulfilling their goal. One of the most common architectures for rational agents is the BDI model, comprised of Beliefs, Desires and Intentions. According to [49], these are defined as:

- *Beliefs:* The perception of agents regarding their world, including themselves. It is a set of statements representing what the agent believes to be true, though the agent might be wrong or the representation might be imprecise
- *Desires:* A set of statements representing what an agent wants or might like to be a part of, in order to pursue or achieve its goal, though it might be the case where there are desires that the agent never fulfills
- *Intentions:* Set of states that the agent is actively pursuing. It might have come from its own beliefs or it might be external, provided by the environment or other agents

Based on its Intentions, an agent chooses which one to pursue. It then reasons upon how to achieve this intention, forming a plan from the actions that will be taken to achieve the goal. Autonomy is a core concept in designing MAS, and what makes MAS different from other computer programming imperative paradigms. Since agents can choose what to do next, their design does not focus on a routine of execution of tasks, but rather an “engine" to process perceptions, desires, beliefs and intentions in order to decide what should be its next action. For example, if an agent is designed with the desire of moving to a particular location, and there is something impeding it from reaching this location, the agent could reason that taking another route could be a way to achieve its desire, or wait until it perceives that its way is not blocked anymore. Rather than giving the agent every single possible case scenario, we are able to give it tools to reason on its own.

2.1.1 Environments

Environments are consolidated as one of the basic building blocks in the design of MAS. This concept emerged from two different perspectives. The first perspective states that environments model the external world from the perspective of agents, providing them with perceptions and enabling them to act upon it in order to fulfill their objectives. The second perspective states that environments need not be only a passive target for agents, but specifically designed to be a part of the MAS, and could contribute to improve the overall development of the system [37]. On the first view, the environment is *exogenous* of the development of the MAS, whereas on the second view, it is *endogenous*. The main abstraction in the process of building environments for MAS are called *Artifacts*, as described in [34], and they emulate humans' day-to-day objects. This is the Environment model that is used in this work. Environments are sets of artifacts that agents interact with, through actions (*Operations* available on the artifact) and perceptions (*Observable Properties* available on the artifact, to achieve their objectives. Since environments could play such an important role in a MAS, they should be carefully designed to support agents, providing artifacts that are constructed, shared and used by agents in the execution of the system. This endogenous approach is the one taken in the development of this work: artifacts and environments are crucial in the design of the MAS.

2.1.2 Coordination of agents and Organization

According to [48], coordination is "a property of a system of agents performing some activity in a shared environment". There are three main reasons why autonomous agents should be coordinated in this context [33]:

- Dependencies between agents: Decision made by agents have an impact on the global state of the system
- Meeting global constraints: There are certain conditions that the solution developed by the group of agents must meet to be considered successful
- Limited knowledge and capabilities: Normally, individual agents have a limited view of their world and are not completely capable of performing all the actions that are required for solving the global objective

This work tackles the coordination of agents by an Organization-Based Coordination approach. $\mathcal{M}$oise [23] is an organizational model that considers structural, functional and deontic relations between agents, and it is able to be specified disregarding the individual behaviour of agents. In this model, the organizational specification is designed in three components:

- Structural Specification: Describes roles that agents should play in the system, and the relationship between these roles (number of agents allowed to take that role, authority between roles, division in groups of roles, etc.)
- Functional Specification: Describes plans and goals that agents should achieve, and their specifications. Goals are grouped in missions, which are then assigned to specific agent roles
- Normative Specification: Describes norms and rules that agents should adhere to. Norms define the expected behaviour of agents, by defining which goals agents must achieve regarding the roles they are assigned to. Norms are further explored on Section 2.1.3

During run-time, agents try to commit to missions described by the organization. The Scheme Board is an artifact that checks if the request is compatible with the mission. The Scheme Board is responsible for tracking missions and their fulfillment, pushing agents to pursue their goals and ensuring that actions are taken in the correct order. The Group Board, on the other hand, deals with the formation of groups of agents in the system.

2.1.3 Open Multi-Agent Systems and Regulation

Open Multi-Agent Systems are MAS where the design of the agents are not known *a priori*. In such systems, agents can enter and leave the system freely, interact with other agents or not, and there is absolutely no guarantee that they have an intention to help achieve a global goal. Open MAS are systems where agents do not know the internal architectures of other agents, agents do not necessarily share a global objective or do not have a global sense of utility, but could do so through collaboration, and the behaviour and interactions of agents cannot be predicted in advance, i.e., they have autonomy of actions [2]. Since agents might not share a global goal, this could lead to a greater extent of problems. Regulation Models in MAS try to solve these problems.

A usual way to solve regulation in MAS are Norms [5]. Norms define the expected behaviour of autonomous agents. Many normative models have been proposed along the evolution of the design of MAS, but in general, they take into account what agents are allowed to do, what they are not allowed to do and what they are obliged to do, given a certain scenario. For example, in the well known normative model ADICO [17], norms are expressed as:

- **Attribute:** Defines which agents have their behaviour regulated by the norm, either specifically ('agent John') or assigned to a role ('agent judge'). They can also be targeted at specific agents playing a specific role or a group of agents;
- **Deontic Operator:** Defines the expected attitude regarding some behaviour. For example, agents might be obligated to, forbidden to or allowed to perform a particular task;
- **Aim:** Describes what should be the outcome of an agent following the norm. For example, the aim for a norm regarding an agent with the role of Judge might be 'produce a verdict';

- **Conditions:** Sets the circumstances under which the norm applies or is valid. For example, norms could have the conditions “upon request of two disputing agents", “based on the rules" or “in five minutes or less";
- **Consequences (Or-Else):** Defines what should happen to agents when the norm is violated or not fulfilled, for example, “or else, agent judge loses judgeship".

Based on the examples provided above, one could create the norm “Agents with the assigned role of Judge should produce a verdict following the rules, under five minutes, when two agents request a dispute, or else it loses the role of Judge". Though this is a superficial description of Norms in MAS, this work closely follows the approach in [8] (explained in more details on Section 2.2) regarding regulation via norms, which also further explores the theory of norms in MAS and comparisons among different normative models.

2.2 Artificial Institutions

Regulation, in MAS, can be contextualized in Artificial Institutions, a concept inspired by the theory of Social Reality proposed by the philosopher John R. Searle in his work “The Construction of Social Reality" [41]. It is grounded upon the idea that we, as humans on a society, often agree and abide to rules that do not represent any concrete concept. We have a mental concept for common objects and their properties, such as “door" or “rock covered in snow", and these constitute the **brute facts** of our world. That is, *brute facts* are indisputable statements about the world, something objective that represents precisely what is and what isn’t. At the same time, we follow rules that describe and structure Institutions that do not have a physical meaning *per se*: money, marriage, government, president; these are all intangible human-made concepts that dictate our behaviour when interacting with others. As we grow, we learn how to live and interact with these institutions, receiving feedback from society, either positive or negative, when our interaction differs too much from what is expected. The author’s argument is that some brute facts *count as* **institutional facts** that base our institutions, and these institutions regulate our behaviour through permissions, obligations and prohibitions. For example, a traffic sign, which is merely a junction of metals that emits light, *counts as* a regulator for vehicles traffic, through lights green, yellow and red. People in vehicles are forbidden to pass the traffic sign if it signals red, and they do so because they abide to human-made rules we call the Traffic Code (which they agreed to follow in order to be allowed to drive in public), not because the red light actually physically prevents vehicles from proceeding. Drivers’ fear of punishment, or hopefully a regard for other humans’ lives and their own, makes them stop at red lights.

The question is: If it is possible for humans to abide to such structures, and oftentimes we do that blindly so, is it possible to create such an institution for the regulation of autonomous agents? For example, is it possible that an autonomous vehicle would stop at a red light, on the sole reasoning that a red light **counts as** a regulator stating a rule that should be followed? Such a behaviour is possible if the agent (the vehicle in the example) acts within an institution that links the traffic lights to some expected behaviour. According to the theory of Social Reality, that is possible if there is some Institution that assigns some value to the traffic lights, so that they get some social meaning that is not given solely by their physical characteristics. According to Searle's theory, institutions are based on some elements, including: *status functions, deontic powers* and *constitutive rules*.

According to [18], Deontic Powers are obligations, duties, authorizations, etc. that shape the expected behaviour of individuals in a society. Status Functions are "functions that environmental elements perform independent of their physical virtues" and Constitutive Rules assign Status Functions to Environmental Elements. Constitutive Rules take the form X **count as** Y in context C. When Status Functions are assigned to Environmental Elements, individuals (who are part of this Environment) have a motivation or drive to behave as prescribed by the norms even though they are able to do the contrary (for example, because a fence counts as some property limit, people are forbidden to cross this fence even when they are able to do so). The work of [8] brings all these concepts together to create a model for Situated Artificial Institutions (SAI), and will base the theory regarding regulation of agents' societies in this work. The next subsections will explore the definitions of these concepts, to clarify what these Institutions are and how they are formed.

Environmental Elements

Usually in MAS, the Environment is considered the set of Artifacts that Agents perceive and act upon. But, from the point of view of the Situated Artificial Institution, the Environment also comprises these Agents and Events happening on the MAS. The Environment is a set of Agents, a set of Events and a set of Properties used to describe the States of this Environment. The formal definition of the Environment is $\mathcal{X} = A_\chi \cup E_\chi \cup S_\chi$ where A_χ is the set of agents possibly acting in the system, E_χ is the set of events that may happen in the environment, and S_χ is the set of properties used to describe the possible states of the environment. These environmental elements carry *status functions* that describe their roles in the Institution.

Status Function

In the original theory, [41] describe status functions as functions that environmental elements perform from the perspective of the Institution. For example, an agent might have all the capabilities for being a traffic ruler and might even perform some functions as one, but without having the status function of *Traffic Ruler* assigned as a Status Function, their actions might not be considered official or they might be prohibited of performing those actions, receiving sanctions otherwise. Only when assigned the role of *traffic ruler* their actions will be considered official to the Institution. The Status Functions are the set $\Phi = A_\phi \cup E_\phi \cup S_\phi$ where A_ϕ is the set of agent-status functions, E_ϕ is the set of events-status functions and S_ϕ is the set state-status functions. A_ϕ is assignable to the set A_χ, E_ϕ is assignable to E_χ and S_ϕ is assignable to S_χ accordingly.

Constitutive Rules

Constitutive Rules specify how Status Functions are linked or assigned to Environmental Elements. They make the specification, based on Status Function assignments, that will effectively regulate the behaviour of agents in the system and make sure its goals are achieved. K is a set comprising all Constitutive Rules κ. κ is a quadruple $\{x, y, t, m\}$, meaning that $x \in \mathcal{X} \cup \Phi$ **counts as** $y \in \Phi$ when event $t \in E_\chi \cup E_\phi$ happened, triggering the Constitutive Rule, and while m holds.

One simple example of these concepts is shown in Figure 1, from [8]: when a particular event is detected (for example, a fire) that **counts-as** an *Emergency* (Environmental Event $t \in E_\chi$), an agent that **counts-as** *Mayor* (Status Function $f \in A_\phi$) *Commands Evacuations* (status function $s \in E_\phi$). These are all linked to the norm: "The mayor is obliged to command evacuation when an emergency occurs". This norm will be satisfied when the agent that counts as Mayor produces an action that counts as Evacuation.

The work of [8] "proposes a model of institutional reality to base the regulation of agent societies, defining its representations as well as its dynamics, that is constituted from the dynamics of the environment". This work also defines a language to design the institutional reality and how the norms base their regulation on an institutional reality. The "final product" of the work is a normative model called Situated Artificial Institution - SAI. The overview of this model is presented on Figure 2. Figure 3 presents yet another example of the SAI Model, modelling the regulation of an Auction, where *Bob* counts as *winner* of the auction, and is obliged to pay for his offer.

The status functions are assigned to, and removed from, elements of the environment as the state of this environment changes, through the interpretation of the constitutive rules. This dynamic is detailed in [18].

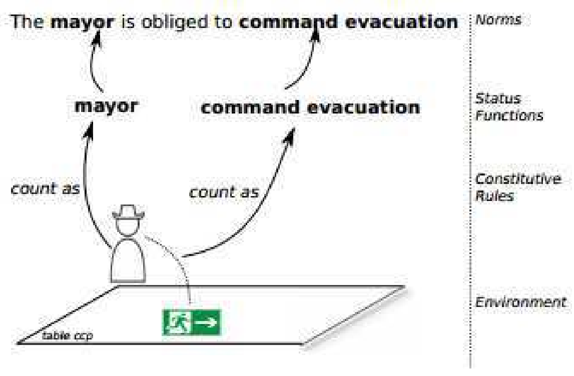


Figure 1 – An example of an Artificial Institution

2.3 Bitcoin and fundamentals of Blockchains

The Blockchain is a technology that received massive attention when it enabled the first digital currency to achieve mass usage, the Bitcoin. Today, it is one of the most discussed

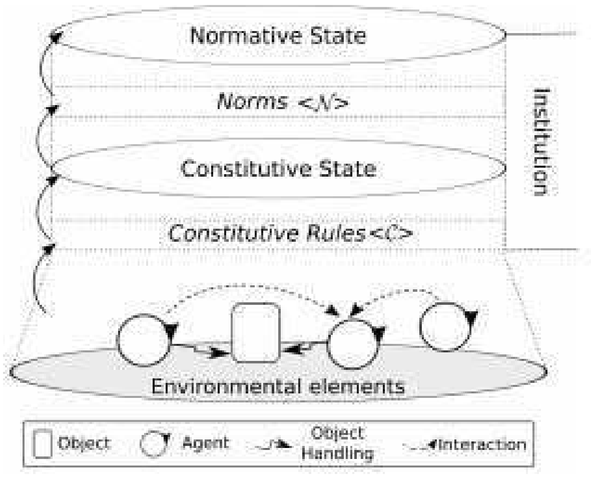


Figure 2 – Overview of the SAI Model

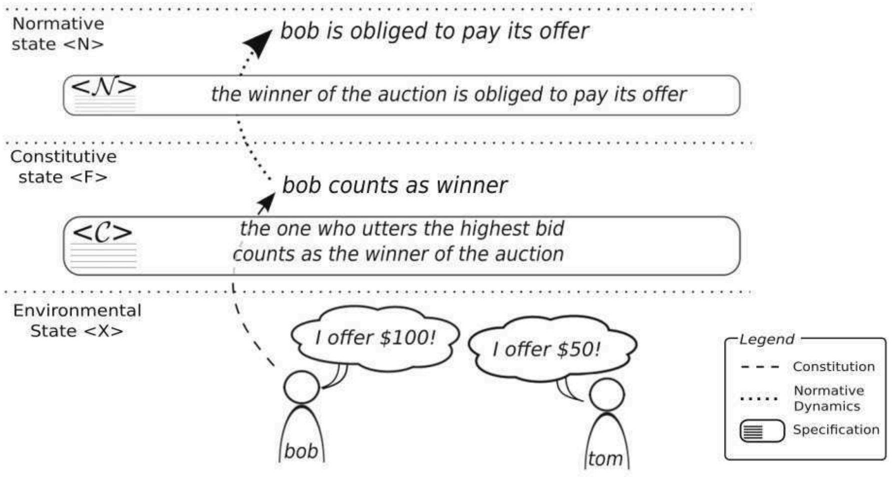


Figure 3 – Example of the SAI model - an Auction

topics in the intersection of finance and technology [43]. There is so much to be discussed about how Bitcoin got to the point where it is and what kind of innovations it created, but for this work it is much more useful to state and understand the properties that a Blockchain provides for applications built upon it. Though Bitcoin's blockchain was not adopted in this work, it is important to look at its fundamental concepts to understand how this technology works. Later, the actual Blockchain that was used in this work - Ethereum - will be explained with further details and its usage justified.

2.3.1 Basic concepts of Bitcoin's Blockchain

The work [32] informally defines the “block chain" as “a ledger in which all Bitcoin transactions are securely recorded". The basic foundation of a digital currency (for currencies based on blockchains, the term that has been used is *cryptocurrency*), like Bitcoin, is that any existing unit (also called tokens or coins) should not be spent twice. That is guaranteed when we argue about physical money: when someone gives a bill to pay for a product, they no longer own that physical piece of paper, thus being impossible that they spend it again on another product. But when that “piece of paper" is digital, it would not be hard to pass it on as payment for a product and then spend a copy of it later. This is called the double-spending problem. The Bitcoin network successfully solved the double-spending problem by creating a common distributed ledger that tracks every transaction (i.e, every passing of a token and its subdivisions from one account to another). This ledger is distributed along to any person with a computer and access to the Internet who wishes to participate in the network. Computers executing this ledger's protocol are called *nodes*. Whenever a token would be spent twice, the network rejects that transaction. Bitcoin is not the first digital currency, but it was the first to become viable and gain popularity for combining three main ideas (that make it a powerful substitute for money, reserve of value and digital transactions system):

- Stiff inflation control: Computational puzzles regulate the speed of creation of new tokens;
- Scarcity: A hard issuance cap that limits the total number of units to 21 million;
- Security and anonymity: Cryptographic signatures ensure that no token is ever spent twice and that consensus over transactions is achieved even when the network cannot know who is sending bitcoin and who is receiving.

The blockchain can be understood as a database that has all of its states' modifications registered and securely signed to verify their validity and veracity. The transactions' data is packed up in blocks that will be linked by hash[1] pointers pointing to the previous block of data. In Bitcoin's protocol, the cryptography algorithm used is the SHA-256, developed by the National Security Agency of the United States, which produces a hash of 256 characters. The most common data structure being used today is the Merkle Tree [32, 44].

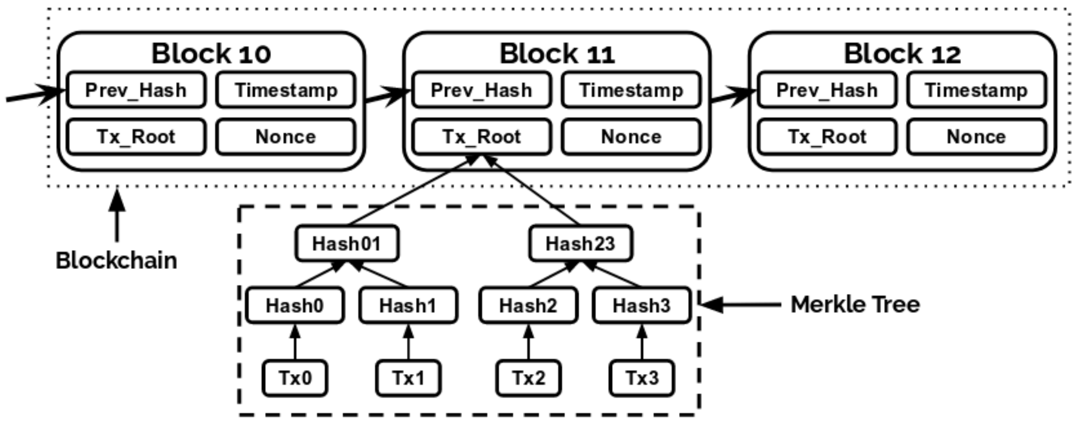


Figure 4 – Schematic of a blockchain

Basic architecture of a blockchain

Figure 4 shows the basic structure of a blockchain. Each block contains a Merkle Tree, which is a linked tree of hashes of transaction data in that block, along with a pointer that

[1] This work assumes that the reader is familiarized with basic cryptography concepts, such as hash functions and digital signatures, and their properties (determinism, fixed output length, collision resistance, signature verification, and so on). There is a vast literature available on this topic, such as [40]

points to the previous block. The block has a timestamp and a Nonce, which is a number that is generated only once in order to approve the block in the network. Finally, it has the hash of the previous block. This is the main feature of blockchains: in order to modify data in an older block, it is necessary to change all the hashes of all the subsequent blocks to maintain consistency. It is easily verifiable that a network which has been tampered with has the last block with a hash that is different from what it was supposed to be. Therefore, one can attest the validity of the whole chain by just looking at the last block's hash. In the case of Figure 4, changing data on block 10 means that the hash of blocks 11, 12, and so on up to the last block, would need to be updated as well. These changes need to be replicated throughout the network, they need to be validated again since the blockchain is a peer-to-peer technology. In that case, only someone with control of more than half of the network's nodes would be able to propagate such changes. This is the reason why blockchains are considered fraud proof. The next subsections will explain how Bitcoin works in more details.

Transactions

Transactions are the basic functionality of Bitcoin's blockchain. They take advantage of Asymmetric Cryptography (also known as Public Key Cryptography). A Bitcoin "account" is a tuple {Public Key, Private Key}. The Public Key is analogous to a bank account number and is called an Address. The Private Key is analogous to a document used to verify the ownership of the account (though your Private Key must not become public at any cost). Whoever knows an address' Private Key, controls that address. The account also contains a list of Inputs, which are the elements that represent coins in the network.

To illustrate the mechanics of a Bitcoin transaction, let's imagine that Alice wants to send 2.0 bitcoins to Bob. Also, let's imagine that Alice can have any number of pieces of paper with any bitcoin value on it, and an unique series number written on each of the papers linked to her address. Alice will tell the network that she wants to use one piece of paper valued at 1.5 bitcoin with series number 'A' and one piece of paper valued 1.0 bitcoin with series number 'B'. Now, since the summed values of these papers (the Inputs used in the transaction) is greater than the desired amount to send, Alice will tell the network:

- "create a piece of paper valued 2.0 BTC and send it to *'BobsAddress'*. The hash value of this address is *'HashOfBobsAddress'*.";
- "create a piece of paper valued 0.49 BTC and send it to *'AlicesAddress'*. The hash value of this address is *'HashOfAlicesAddress'*.";
- "to whoever finds this message, create a piece of paper valued 0.01 and keep it to yourself as a fee.

Alice will then sign the pieces of paper she is using for the transaction with her Private Key. The node on the network that finds the message will then take Alice's Public Key and verify if this

Public Key is paired with the Private Key used to sign the pieces of paper, meaning she owns them. It will turn to the ledger and check if papers ‘A’ and ‘B’ have not already been used. If not, then it will go to Bob and say: “I have this paper valued at 2.0 bitcoins to deliver to you, but you must prove that you are the intended receiver. Please, use your Private Key to verify that you are the owner of this address.". If Bob can verify that he is the owner of the address, he will receive this new paper with 2.0 bitcoins and series number ‘C’. The node will do the exact same procedure to deliver the 0.49 bitcoins back to Alice, with series number ‘D’, even though she was the one creating the transaction. It will then go to the ledger, the database, and state: “Transaction ‘AliceToBob’ has happened. Pieces of paper with series number ‘A’ and ‘B’ have been used in this transaction. Pieces of paper with series number ‘C’ and ‘D’ were created. All of this happened at this exact moment".

The new “pieces of paper” that were created and delivered to Bob and Alice, and the fee to the node performing the transaction, are called Outputs of the transaction. An Output of a transaction to an address will be an Input of a future transaction from that address. Alice’s account now has three Inputs: 2 that are ‘invalid’ summing up to 2.5 bitcoins, and one ‘valid’ with value 0.49. Her total balance is 0.49 BTC. Bob, on the other hand, has one valid Input of 2.0 BTC. If Alice now tries to use Inputs ‘A’ or ‘B’ (meaning she tries to double-spend them), the node on the network will look them up at the ledger to check if they were already used and will reject the transaction, avoiding the double-spending problem.

It is important to notice that the node validating this transaction actually has absolutely no idea who Alice or Bob are. It is just receiving instructions from an address to deliver to another address, which are in fact random-looking long strings (like ‘1P5855cfMdUnVMFocFT2h. . . ’). Bob might actually have no idea that he is about to receive a transaction from Alice, and he can never be certain that the transaction came from Alice unless he knows that that address belongs to Alice. He only knows that he received a transaction from the address *‘AlicesAddress’* (or, more realistically, ‘13DhL8T7y2bdRJAQ5eTp6BXzCzVLm3va3z’).

This is a very simplified explanation of a Bitcoin transaction, because the more intricate details are out of the scope of this work. The work of [32] is filled with all the details and nuances relating to Bitcoin transactions.

Block Creation and Mining

Nodes include blocks of transaction by spending an enormous amount of computing power to solve a computational problem that can only be solved by a *brute force* algorithm. A node that wants to add a block to the blockchain will take the previous block hash, the data from all the transactions contained in the block (senders’ addresses, receivers’ addresses, inputs, outputs, timestamps, etc.) and a number that it randomly generated. It will then append all this information together and produce a hash out of it. This hash has to meet some specific property that satisfies a requirement that severely limits the domain of possible outputs of the hashing algorithm. For example, this requirement might be that this hash ends with 3 zeroes. This is called the *difficulty* level of block generation. Finding a hash that ends with 3 zeroes is easier than finding one ending in 4 zeros, and harder than finding one ending in 2 zeroes. The node will almost certainly need to test a large number of randomly generated numbers until it finds a solution. The number that makes the final hash meet those requirements is the **Nonce** of the block.

There is a big reward for spending vast sums of electricity in this computational effort (which is called Proof of Work - PoW). This is exactly the point where the creation of new units of the currency happens. The newly created coins are the reward for that node. When a node solves the puzzle and confirms a block, this block is then appended to the blockchain and propagated through the network. This process of creating blocks of transactions, and subsequently creating new coins, is commonly called *mining* the currency, and the nodes that do this PoW process are called miners. Other miners, who are competing to be the next to find a new Nonce and append a block to the chain, are able to easily verify if the Nonce proposed by the winning miner does solve the puzzle and if the hash of the last valid block that they have was indeed used to create the new block (attesting that no data was changed in older blocks).

There are many sources that go deeper into the mechanics of Blockchains, such as [1], and again, [32]. But this simplistic explanation should be sufficient to understand why Blockchains are considered fraud proof. A malicious node that wants to append wrongful information to the chain needs to spend a massive amount of computational effort to find the Nonce, and then control at least 50% of the remaining miners (which are also spending massive electricity) to confirm that the new block is valid. When a block is added to the network and propagated to other nodes, there is effectively different versions of the blockchain in different nodes. Because of that, recently added blocks cannot yet be considered trustworthy, because it takes some time for the network to achieve consensus over it. Users of the network wait for some new blocks to be added after the specific block they need to trust. In Bitcoin's case, it is common to consider a block immutable after 6 other blocks are added. When that happens, the block and the transactions on it are said to be “confirmed". Since Bitcoin's ledger grows at about 1 block per 10 minutes, one normally waits for about 1 hour before being sure that a transaction is valid on the network.

Mathematically speaking, tampering with the data in a blockchain is indeed possible, but the odds of modifications propagating through the network, in the present state, are so small (such as the odds of the planet Earth being hit by a large meteor in the next 3 seconds [4]) that it is considered impossible to commit such a fraud. That happens because the amount of work needed to mine a new block is extreme. It is so computationally expensive to mine new blocks that specific pieces of hardware were designed for the task. Common computers were very quickly unable to do the job, lacking processing power. Figure 5, taken from [22], provides a simplified, yet fairly accurate, representation of a transaction in Bitcoin's blockchain:

The main takeaway from this is, again, that this distributed ledger is considered fraud-proof. That is the cornerstone of the “faith" in the network. Gold has been used by humanity as a reference of value for millennia because it is useful to humans, the process of taking gold

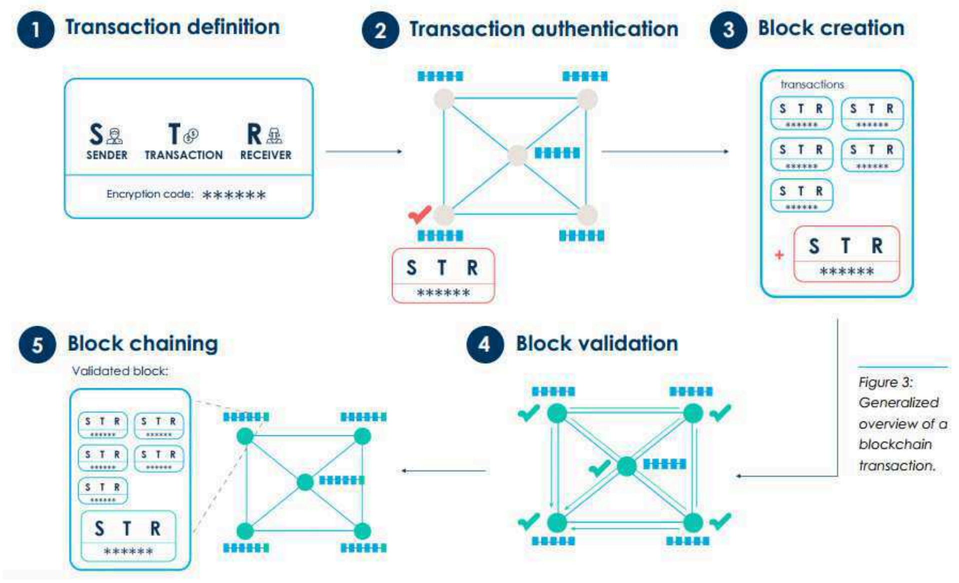


Figure 5 – Schematic of a Bitcoin transaction, extracted from Evry's Blockchain Whitepaper

from the ground is very hard (therefore, slow and expensive) and it is impossible to counterfeit gold. Gold is useful, has a stiff inflation control and a hard issuance cap. Bitcoin can be used as a reference of value because it will eventually stop being created just like gold (which means no real inflation, and eventually, deflation), and because currently there is no computational power available to fraud its ledger of transactions. For that matter, a blockchain allows consensus without the need of trust between nodes, and without the need of a central power or arbiter. This concept is important to clarify why computation done along the network is trustworthy.

Distributed Consensus

As illustration of this distributed consensus, let us think of a voting system. You write your vote on a piece of paper, sign it with your secret key, and write down a public key (a key to perform a cryptographic operation over some information) on a central book. Your piece of paper **can** be found, but this public key is absolutely anonymous. In order to make your vote valid, someone will take your public key, perform the cryptographic operation on it to verify it was you who voted, and check that your vote is valid (for example, check if you have not reached the limit of times that you are allowed to vote). Better yet: upon validation, the previous vote is checked and confirmed to be valid, meaning that all votes up to that point in time are valid. This information is added to your vote, your vote is signed and confirmed to be valid and the vote is computed. When that is done, only the last tuple {public key, vote} needs to be checked for audition of the whole voting system. That is why the blockchain allows consensus (all votes up to this point are trusted) even when trust is not present in the network (the nodes do not know who is voting, therefore they cannot trust it *a priori*).

As stated before, the Bitcoin blockchain was not adopted in this work, because it only supports one operation: transactions from address A to address B. There are different Blockchain protocols available, with more advanced features. The concept of the cryptocurrency has been extended to the concept of *Smart Contracts*: a digital contract written on a blockchain will **always** behave exactly as it was programmed and published [12]. This contract is based upon atomic operations: either its functions will execute given a condition or trigger, or not execute at all. On the next section this concept is explored with more details.

2.4 Smart Contracts

The Blockchain quickly evolved from a secure record of transactions to a platform for distributed computing, in the form of Smart Contracts. The Ethereum Project [12] uses the computational power from the nodes of its protocol for decentralized computing of functions. That is, one can deploy computational functions that will give outputs to determined inputs as long as the blockchain itself is executing in the decentralized network. Nodes will be rewarded to perform the execution of these functions. These pieces of code are the Smart Contracts. Taking advantage of the nature of blockchains, Smart Contracts are immutable, where only parameters of the operations can be changed by participants and only when there is consensus about it. If two nodes agree upon a certain contract, there is the absolute guarantee that it will execute when conditions are met. A simple example is a contract that will transfer a particular amount of tokens from participant A to participant B when a bar code is read, meaning participant A bought something from participant B. Participant A will not be able to default on the agreement with participant B, and if there are no funds in the account, then the sale is never completed in the first place. Participant B could, for example, have the contract check if Participant A has the necessary funds deposited on the contract before delivering the product. This “either wholly do some specific action, or don’t do it at all" is called an Atomic Operation, and is a powerful characteristic of Smart Contracts.

This particular concept of Smart Contracts could be very valuable to MAS applications regarding integrity of actions, because it leaves no ground for dispute. Currently, the Bitcoin protocol does not support Smart Contracts. The most developed platform for this technology today is the Ethereum Project, which was used to develop the Smart Contracts used in this work. ‘Under the hood’, Ethereum’s blockchain works quite differently from Bitcoin’s. For example, Ethereum’s hard cap on currency limit is much softer (with proposals to extend it to 120 million units [11]). But looking from a higher level perspective, which suffices for this work, the main and crucial difference is indeed the ability to run Smart Contracts. All other characteristics and advantages of Bitcoin’s Blockchain are also present when using Ethereum.

2.4.1 Ethereum Virtual Machine and JSON-RPC Protocol

Ethereum is built upon the JSON-RPC protocol, which stands for JavaScript Object Notation - Remote Procedure Call. It is a stateless, lightweight data-interchange communication protocol and data format, that can represent numbers, strings, ordered sequences of values, and collections of name/value pairs [29].

Ethereum can be thought of as a worldwide computer. It has a virtual machine called the

Figure 6 – Ethereum Project's Logo

EVM - Ethereum Virtual Machine, a Turing Complete machine that operates much like traditional computers or processors: it pushes and pops instructions executing the Smart Contract's code, transitioning from one state to another and producing outputs that are consumed by participants on contracts. This EVM executes a low-level bytecode, consisting of a series of bytes representing stacked operations. These operations access three types of memory spaces to store data:

- A stack, last-in-first-out data structure where values are pushed and popped;
- Memory, an infinitely expansible byte array;
- A long term storage, that works as a key/value pair storage; this storage persists for the long term, unlike the stack and the memory

Whereas in Bitcoin's protocol nodes receive instructions to make transactions from one address to another (as described in subsection 2.3.1), in Ethereum's protocol the node will receive a request to perform operations over the n-uple {*block_state, transaction, message, code, memory, stack, program_counter, gas*}. The node will then take the *program_counter*-th byte of *code*, modify the *global_state* according to the operation being executed (for example, summing two variables, modifying a string or transferring funds from one account to another), doing so in an infinite loop until *program_counter* > *len(code)*. The modified *global_state* is now the new state of the network (analogous to the transaction in Bitcoin, which modified the state of the network), propagated for validation. The variable *gas* dictates the "price" the owner of this code has to pay in order to have it executed.

The EVM provides a high-level programming language called *Solidity* to program and compile the bytecode used in Smart Contracts. There are more details about the EVM and its execution available at the official documentation website [20]. The main takeaway is that the Ethereum Protocol is a blockchain-based distributed computer. Operations performed on this computer can be trusted to be true, just as transactions on Bitcoin's protocol. The outcomes of these operations will be immutable and stored as long as the network is running. Developers can write very complex code and deploy to this network, enabling even more higher-level features for their applications. Virtually anything can be represented on a Blockchain of this nature, as long as any characteristic of the real object can be digitized [43].

2.4.2 The “value" of Ether

It was previously stated that value is a subjective concept. The value of Ether (or any cryptocurrency, or anything really) as a currency or money can be easily understood from the perspective of the Austrian Economics school of thought. One of its most notorious exponents, Ludwig von Mises [46], explains that ‘money’ can arise or be defined by two sources: forces of the market, that is, people engaging in free trade of assets, or from a central power by decree and monopoly of currency issue. An analysis of cryptocurrencies as money from the second perspective is utterly useless, because it is notorious that there isn’t any central government forcing people to use them as money, as they provide their own monopolistic nationalized currency. Dr. Friedrich von Hayek has also provided theories on the subject of money, specially in [45].

According to the first perspective, currency is a commodity or product like any other. Something becomes money when it fulfills some specific requirements: it must be regarded as a means of exchange without being consumed. For example, when someone exchanges a loaf of bread for chicken, it is expected that those products will be consumed at some point, otherwise they will deteriorate and ultimately lose all their usefulness. Chicken and bread are terrible candidates for money. A piece of paper stating that you own a certain amount of wealth can be exchanged without the fear of it losing its value overnight, supposedly. Though this is a necessary condition, it is not sufficient. Money must fill some other requirements: it must be scarce in order to hold value, it must be spatially and temporally homogeneous (under the same society or context, it must amount to a similar purchasing power in proximate locations and dates), it must be divisible up to some point, transportable, etc. Those are characteristics that make things a better or worse kind of money. A currency that is constantly debased in regard to another (through inflation, for example) is a currency of inferior quality, because it can be exchanged for fewer goods and services over time. Though it might have the same number printed on it forever, it leads to less consumption for its owner in the future. But none of these actually determines the ‘price’ of money. What determines the price or value of money is the open market. That is, what people are willing to exchange for that money, with the perspective of potential consumption of products with it in the future.

To illustrate this concept, let us think of a baker that might be willing to exchange 10 pieces of bread for 100 units of a particular currency today. Tomorrow he might only be willing to exchange 10 pieces of bread for 110 units, because the baker fears or expects that 100 units of this money will not be sufficient to get the same amount of products he gets today (granted, this is not the only factor that makes the baker change the price for his bread, but it is an important one).

The subjective value perceived by potential users of this money is what drives its 'price'. From the perspective of the buyer, 10 pieces of bread costs him 100 units. From the perspective of the seller (the baker), 100 units of this money (which he needs in order to buy flour to make bread tomorrow) costs him 10 pieces of bread. That is the reason why anything has a price relative to another thing. 'Money', whatever form it takes, is just 'oil in the gears' of daily transactions, a facilitator. Similarly, Ether has a price in bitcoin, in American Dollars, in Brazilian Reais, in chickens, in petroleum, in olive oil ... (even if that price is 0, that is, people are not willing to exchange anything for it).

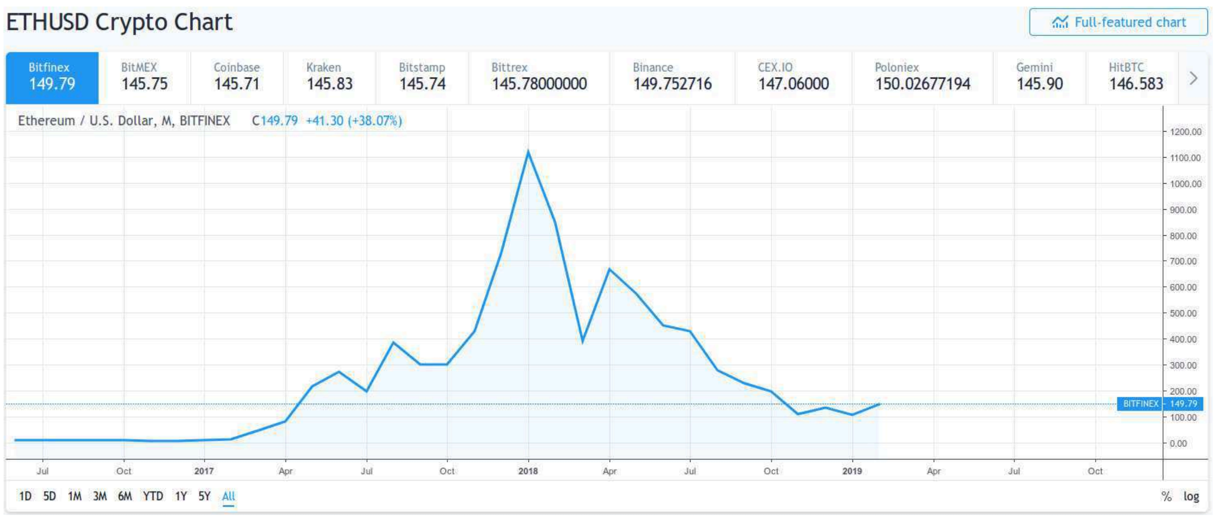


Figure 7 – The historical price of Ethereum in American Dollars on 23/02/19.

Though leading cryptocurrencies like Bitcoin and Ether fill most, if not all, of the requirements of good money, the objective of this subsection is not to vouch for them as such. The objective is to show that even something completely virtual, like cryptocurrencies, can and are regarded as money by some people, because of the aforementioned characteristics. This clarifies that cryptocurrencies are assets and, on a worst-case scenario, work as proxies for real world, physical money, which leverages their use in Multi-Agent Systems. Figure 7 shows the historical price of Ethereum in American Dollars.

2.5 Literature Review and Related Works

The Blockchain is a relatively new concept with about 10 years of popularization since the conception of Bitcoin. The theory of Blockchain and its applications are covered in details in [31], [32] and [1]. Improvements and applications are occurring mainly outside of the academia, but it has drawn larger interest of researchers in the computational and economical fields over the last couple of years. The topic has recently gained peer-reviewed journals, such as:

- *The Journal of British Blockchain Association* (JBBA) - "Europe's first peer reviewed, academic journal devoted to Blockchain technology, Distributed Ledger Technology (DLT) and Cryptocurrencies" [3] ;

- *Frontiers in Blockchain* - “Frontiers in Blockchain publishes rigorously peer-reviewed research covering all theory and applications of blockchain and blockchain-related technologies" [21] ;
- *The Ledger Journal* - “a peer-reviewed scholarly journal that publishes full-length original research articles on the subjects of cryptocurrency and blockchain technology, as well as any relevant intersections with mathematics, computer science, engineering, law, and economics." [27] .

The work of [51] gives a systematic review of research on Blockchain by 2016. It states that about 80% of papers were research of improvements for the Bitcoin network itself, mainly security, wasted resources and scalability. The other 20% is research on Smart Contracts, privacy in the network and other cryptocurrencies. Using the aforementioned journals as reference, it is possible to have an idea about the current research on Blockchains as well. Over these three journals, there are 37 peer-reviewed articles. They can be categorized as:

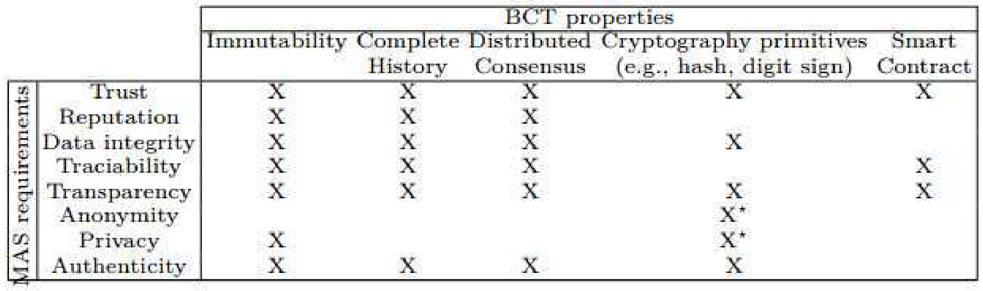
- **Theory, Algorithms** & **Improvements** Articles covering theoretical aspects of Blockchain: 12 out of 37, 32%;
- **Economics and Market Analysis** Articles covering economical application and market analyses of cryptocurrencies, 9 out of 37, 25%;
- **New Tokens** Articles covering proposals and analysis of new cryptocurrencies: 4 out of 37, 11%
- **Applications** Articles covering applications of Blockchains to areas that do not involve finance or economics: 12 out of 37, 32%.

IEEE Xplore has also shown a significant increase in publications regarding Blockchain. A simple search on the website `https://ieeexplore.ieee.org/` shows over 1,650 entries for the subject by March 2019. From 2013 to 2016 there were 402 entries. From 2017 to March 2019, 1,280. The results show that works published at IEEE Xplore range from very complex and specific applications to very general papers explaining the Blockchain Technology. It would require a much more thorough investigation to assess all of these works, but we can draw an idea about what is being researched in the intersections of Blockchain and Artificial Intelligence and Multi Agent Systems. A search with the terms “Blockchain Artificial Intelligence” returns 98 entries, and another search with the terms “Blockchain Multi Agent Systems” and “Blockchain Agents" returns a combined number of 52 results, obviously with some overlap.

Looking specifically into the intersection of Multi-Agent Systems and Blockchain, there are a few related works that tackle similar problems or that could benefit from some of the results achieved in this work. In *“Blockchain-based protocol of autonomous business activity for multi-agent systems consisting of UAVs"* [26], the authors model and propose “an architectural solution for organizing a business activity protocol for multi-agent systems.”, which has a clear link to the model for supporting transaction of assets presented in this work, specially the regulation of the system by the SAI. There are other two works, from mainly the same authors, *“Blockchain Based Protocol for Economical Communication in Industry 4.0"* [25] and *“Robonomics Based on Blockchain as a Principle of Creating Smart Factories”* [24] that deal with the same broader problem of autonomous agents transacting assets.

Looking at broader aspects of MAS, there are works proposing solutions for Coordination of Agents [16], Negotiation Models [13], Accountability [36], and applications of these two technologies integrated to solve problems, such as [10]

[14] provides a more structured literature review about MAS and Blockchain, by reviewing publications and providing an “overview of their domains, requirements of the application scenarios, motivations, assumptions, strengths, limitations, and identification of future challenges". This work provides a useful table [2] that compares Blockchain Technologies properties (called BCT by the authors) to MAS requirements, shown in Figure 8.



		BCT properties				
		Immutability	Complete History	Distributed Consensus	Cryptography primitives (e.g., hash, digit sign)	Smart Contract
MAS requirements	Trust	X	X	X	X	X
	Reputation	X	X	X		
	Data integrity	X	X	X	X	
	Traciability	X	X	X		X
	Transparency	X	X	X	X	X
	Anonymity				X*	
	Privacy	X			X*	
	Authenticity	X	X	X	X	

Figure 8 – A table comparing Blockchain properties and MAS requirements, taken from [14]

[2] * The requirements of Anonymity and Privacy require off-Blockchain Cryptography primitives, that is, the sole employment of a Blockchain solution does not guarantee Anonymity and Privacy.

3 Conceptual Model and System Design

The objective of this chapter is to discuss and propose different models for Multi-Agent Systems integrated with a blockchain and argue over which would be more suitable for implementation. This chapter proposes this new conceptual model for MAS where blockchains are inserted and propose a possible solution to a suitable problem in the domain of MAS.

Both Blockchains and Multi-Agent Systems are applicable to problems that have the common feature of information decentralization or computational distribution, such as Energy Grids, Supply Chain Management, Marketplaces, Economic Modeling, etc. Even though security is one of the motivations for the usage of Blockchains in MAS, it will not be the primary concern of this work, but more of a welcomed “side effect". More specifically, the objective of this chapter is to establish a link between these two technologies, to understand and evaluate what is the best fit for a Blockchain in a MAS, specially taking into account some disadvantages that Blockchains may present.

3.1 Conceptual models for MAS and Blockchain

It was previously stated that the initial hypothesis is that the Blockchain has a role to play when we develop a MAS that features transaction of assets among agents. The starting point of [8] is the “design of artificial institutions that not only specify the regulation but that also have means to specify the institutional reality on which the regulation is based, arising from the concrete environment where the agents act.". Therefore, the goal of this chapter is to outline the design of a suitable model for a MAS regulated by Situated Artificial Institutions, where Ethereum’s blockchain, via Smart Contracts, bridge the gap between the virtual world of agents and the real world of assets’ transaction.

This work assumes some starting guidelines regarding the components of the MAS. From there, different approaches to a Blockchain model are explored, from which one is selected. This chapter aims to logically walk through possible usages of the Blockchain in MAS, for completeness, in order to settle at the most reasonable choice.

3.1.1 Agents, Environment, Organization and Institution

This work considers that agents are, at least on a first concept for simplicity, composed of the common BDI model, designed and implemented using the Jason framework and programming language [7]. But since Open MAS are being considered, there is absolutely no guarantee of previous knowledge of these agents’ design. The environment is the set of artifacts accessible by the agents to perceive and act upon. It will provide all of the artifacts needed by the agents. This is where, initially, the blockchain will be inserted. The artifacts used in this work come from the

CArtAgO specification [38]. Finally, the Organization follows the $\mathcal{M}$oise specification.

These three models are provided by one single software, a framework called JaCaMo. “JaCaMo is a framework for Multi-Agent Programming that combines three separate technologies, each of them being well-known on its own and developed for a number of years so they are fairly robust and fully-fledged." [6]. With JaCaMo, a MAS can be designed and developed using Jason agents, CArtAgO artifacts and $\mathcal{M}$oise organization, easily integrated.

As mentioned before, the theoretical and practical concepts of Artificial Institutions closely follows the work of [8]. Artificial Institutions were explained in more details on Section 2.2. The language, now integrated into the MAS development framework JaCaMo, will be used to implement the Situated Artificial Institution. The SAI will base the regulation of the MAS.

3.2 Integrating the Blockchain to the system

Today, it is generally accepted that the Environment is an essential part of the development of Multi-Agent Systems. Classic AI not only brings the concept of environment, but also defines agents as anything that perceives, through sensors, and acts upon its environment through effectors, such as the definition brought by Russel&Norvig [39]. In the MAS domain, *artifacts* have been proposed as entities to compose the MAS environment in order to achieve better coordination between agents and to provide them with information about their environment, allowing for enhanced cognition. Artifacts are said to be reactive entities to provide functions that make agents cooperate in MAS, and shape the environment according to the system’s needs [34]. Because the SAI model is used to regulate agents’ interactions, and some of those interactions involve the transaction of values and assets, we need to generate events in the Environment so that the SAI can base the regulation of the system on them. Therefore, we need a model where the Blockchain is connected to the Environment of the agents. At this point, it is important to understand what integrating a Blockchain to the system means, that is, what are the possible drawbacks and unwanted consequences. In the sequence, we discuss possible places for the Blockchain in the environment shared by agents.

3.2.1 Blockchain as a generic environment

The whole environment of the MAS could, theoretically, be modeled through Smart Contracts running on the blockchain. The Ethereum Project is the first and most reliable provider of Smart Contracts and, as stated before, will be utilized in this work. All the environmental objects that agents need or have access to could be coded as Smart Contracts, since they are Turing Complete machines. One big advantage of this approach is the convenience of developing every needed object on the same platform, without the need for extra technology or interfaces. In this case, every single action or transaction made by agents would be recorded and immutable, providing enhanced transparency to the system. Whether an agent is delegating a task to another agent, sending another agent a particular amount of currency or simply stating a fact, everything would be forever recorded, as long as the blockchain exists. Figure 9 illustrates this proposed model, where agents will only have access to artifacts and other agents through the blockchain. In this figure, agents (the human like figures) can interact, via the dotted lines, with some other agents, Smart Contracts that are outside the scope of the MAS (the yellow and green boxes) and the Artifacts, which are the Smart Contracts defined in the context of the MAS.

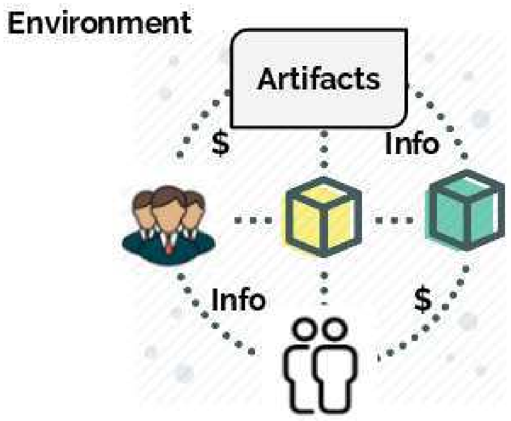


Figure 9 – Model of the Blockchain as a generic environment

Though this might seem ideal, this implementation would not come cheaply. As explained on section 2.4.1, contracts need to pay transaction fees for having the network execute its code. How much a Smart Contract costs really depends on the complexity of the code. Surely, running an $O(n^2)$ code will be more expensive than running an $O(n)$ function. According to the website https://ethgasstation.info/, which provides real time data on the costs of running code on Ethereum, a basic transaction on Ethereum's main network could cost 0.00021 ETH, or US$0.02856 (as of February 2019). Considering a MAS with 200 agents, where half settles debts with the other half every day through one transaction per pair, this Smart Contract would amount to about US$1050 annually, on transactions alone and not considering other computations done on the Smart Contract.

Also, blockchains can be very slow, specially large public blockchains, such as Bitcoin or Ethereum. For example, a simple Bitcoin transaction takes at least 10 minutes to be executed, and could take up to one hour to be considered confirmed. Today, it operates at the speed of about 2.5 transactions per second. Ethereum has faster confirmation time, about 15 to 20 seconds. As stated on chapter 2, after one block is added to the chain, the network needs to wait for some extra blocks to be added after it, so it can be confidently stated that a transaction is secured. In Ethereum's case, users of the network tend to wait for 15 blocks to be added before guaranteeing that a block is valid and immutable. Therefore, users of the network consider a transaction confirmed after 5 minutes, which severely limits the application of Blockchains to systems where speed requirements are crucial.

To add to the drawbacks of using blockchains, scalability is another problem to be taken into account. Scalability is, in fact, Blockchain's main technical issue for mass adoption, regarding computational power, memory for storage and Internet bandwidth. Both Bitcoin's and Ethereum's ledgers were about 200 GB in size by early 2019. As the technology gains adoption, this problem will only get worse [15]. Satoshi Nakamoto's idea on this was that Moore's Law (the idea that computational power on common devices and computers doubles every year) would take care of both processing power and storage needed to run a node of the Bitcoin ledger, as stated in the original proposition [31]. But the large attention Bitcoin has drawn in a decade was not foreseen. With the concentration of mining power in just a few "players" concentrated mainly in China and Russia, the development of the system became slower and more difficult than in the earlier days, specially when there are conflicts of interest between miners, developers and users. This hampers the evolution of the network and the mitigation of those problems.

These factors have to be taken into account when designing a MAS that will rely on blockchain, specially when taking the approach of implementing every single artifact as a Smart Contract. The whole system could become impracticably slow, with every single action performed by agents needing confirmation from the blockchain before being executed.

Because of these drawbacks, the proposition presented above does not make a reasonable use case, specially when we think again of human societies: we do not register every single interaction we have with other humans or objects. We care to register only a small subset of these interactions, normally those that involve promises with serious consequences, contracts, transactions of value and so on. This brings the notion that perhaps a blockchain is better modeled for MAS as one or more separate artifacts available to the agents, and not the whole of the environment, in order to register special interactions that can be used for the regulation of the system.

3.2.2 Blockchain instrumenting application artifacts

As stated before, this work takes the approach of carefully developing artifacts that have a meaningful design as to support agents in their tasks [34]. Since Smart Contracts are programmable contracts that can be developed to perform specific functions, it makes sense to envision these Smart Contracts supporting specific suitable artifacts in the environment, and these artifacts will create the interface between the contract and the system. For example, a voting system to decide whether agent A has endorsed a false statement can be modeled in one Smart Contract to store and officially count the votes that agents have deposited in the corresponding artifact. Artifacts could also implement authorizations for actions, with much more security designed and embedded, for example, a strict method of access available only to agents carrying very specific roles or authorizations. These choices of design really depend on the requirements of the system and its goals. Figure 10 shows how, in this model, artifacts model the interface between agents and the blockchain. Agents are still able to flexibly communicate among themselves, and access other types of artifacts.

The scalability of this model greatly depends on the requirements of the system. The Ethereum Blockchain is essentially a world computer where users pay mining nodes for performing

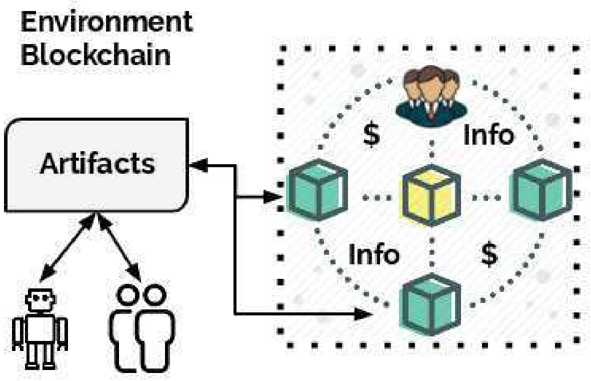


Figure 10 – Model of the Blockchain as specific artifacts in the environment

computation for them. But it is a very slow computer (somewhere around 10 to 30 executions per second). If the system requires the advantages brought by blockchains in all of its artifacts, then it must loose its requirements of speed. The number of artifacts implemented in the blockchain should not necessarily be limited, but there will always be a compromise of execution time and number of interactions with the blockchain. The example of the house building application, presented on Chapter 4, makes this clear: modeling the auctions in separate artifacts running on the blockchain, as in [36], makes the execution of the system orders of magnitude slower than a simple execution of the auction in this MAS by itself, with payments then executed via blockchain.

This integration is done by "translating" the desired artifacts to the Smart Contract language. Since JaCaMo artifacts are Java classes, the easiest way to make this integration is to have Java classes and methods that are able to communicate with the Ethereum Network, that is, a Java implementation of the JSON-RPC protocol. A third-party library called Web3j (available at `https://web3j.io/`) was used to bridge this gap. This library provides Java classes wrappers using the pre-compiled Smart Contracts, written with the equivalent logic of the desired artifacts. There is no specific observable property or operation available on this artifact, being dependent on the problem to be solved. At first, there is an artifact *A* with the necessary specification, operation *operation_A* and observable property *obs_prop_A*. These operations must be coded as Smart Contracts functions, in the language *Solidity*. To port observable properties, the Smart Contract will need to declare a function as *public view*, so that external entities are able to retrieve the required information (for example, the function *checkValue* given on Appendix B, which retrieves the value of a hash map given a key and returns that value), otherwise it is kept hidden by the Smart Contract. The artifact will become an interface of interaction between agents and the corresponding Smart Contract running on Ethereum.

Figure 11 shows an overview of this proposal. In this structure, the Artifacts (yellow boxes with operations and observable properties) *base* the code of the Smart Contract. This Smart Contract, when compiled, *generates* a Compiled Code, represented by a dark blue box on the Figure, which will be *deployed to* the network. The library will also *generate* the Java Wrapper Classes, which are classes that communicate with the coded functions that where deployed. These classes must be, then, *imported into* an Interface Artifact (denoted by the orange artifact). The artifact is a common CArtAgO artifact that is now able, if needed, to read a local Ethereum wallet file, deploy contracts to the network, make transactions, perform operations on the Smart Contract and so on. It will be able to operate and retrieve information from the network, as requested by Agents, through an Accessible Node (which can be running locally or remotely). It will also generate the environmental events that base norms and regulation of the system, by the SAI, which will condition the behaviour of the Agents.

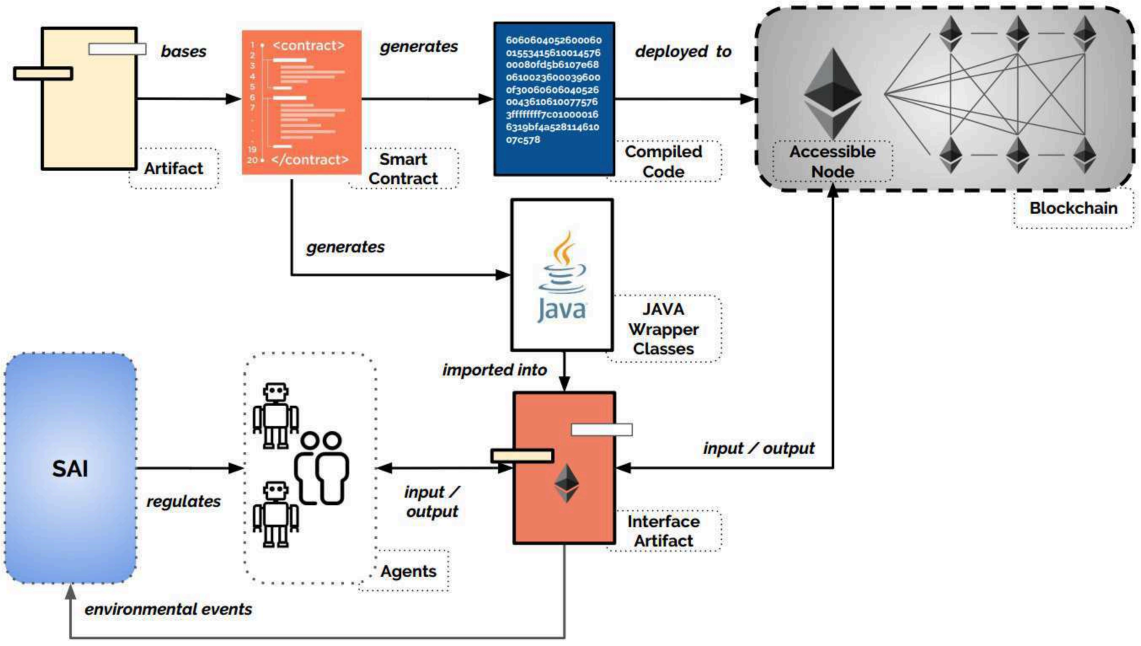


Figure 11 – The whole integration of the system

Interface Artifact and Smart Contract

To illustrate how the integration of these technologies is achieved, [36] provides the implementation of the *Auction Artifacts* (described in Chapter 4) in the Smart Contracts. Consider an artifact with the operation *Bid* and observable property *currentWinner*. The *Bid* operation registers a bid from a participating agent and *currentWinner* states who is the current winner for that auction. The following code is an excerpt that implements these two functions in the Smart Contract:

```
## Function Defined in the Smart Contract -- AuctionMAS.sol##

    # defines the function placeBid #
    function placeBid(string task, uint bidValue, string bidder)
      public {

        # retrieves the auction for task from hashmap auctionList
        Auction storage a = auctionList[task];
        if (a.currentBid > bidValue){
            # updates winning value
            a.currentBid = bidValue;

            # updates current winner
            a.currentWinner = bidder;
        }
    }
...

    # defines the function getCurrentWinnerbyAuctionID #
    function getCurrentWinnerbyAuctionID(string task)
      public view returns (string winner) {
        # public view makes this variable observable

        Auction storage a = auctionList[task];

        # returns current winner
        return a.currentWinner;
    }
...
```

With the above code compiled, the library Web3j is able to generate Java classes that are able to communicate with the Ethereum protocol when this code is deployed to the network (the generated code is of smaller importance to this work):

```
## Auto-Generated Class Wrapper -- AuctionMAS.java ##

    # generates the wrapper for getCurrentWinnerbyAuctionID #
    public Future<Utf8String> getCurrentWinnerbyAuctionID(task)
    {
        Function function = new Function("getCurrentWinnerbyAuctionID",
            task;
        return executeCallSingleValueReturnAsync(function);
    }
...

    # generates the wrapper for placeBid #
    public Future<TransactionReceipt> placeBid(
        task, bidValue, bidder)
    {
        Function function = new Function(
            "placeBid", (task, bidValue, bidder);
        return executeTransactionAsync(function);
    }
...
```

Now, we are able to import these classes into the artifact implementation. This artifact has an operation available to the agents called *Bid*, the same as before, but instead of having the logic of the bidding function, it will manage the interaction of the agents and the network. It needs the address in the network where the Smart Contract was deployed to (*"0x0401..."*), and some supporting variables to control the price it is willing to spend on operations (*gasPrice, gasLimit*). It also needs the access to the Ethereum Node (*web3*), which is a local Http Service. Finally, it needs *credentials* (password and Ethereum Wallet) to access the Smart Contract.

```
## Artifact Code -- AuctionArt.java##

    # loads the wrapper classes #
    import AuctionMAS;

    # connects to the local node #
    private Web3j web3 = Web3j.build(new HttpService());

    # loads Ethereum wallet file and password #
    Credentials credentials = WalletUtils.loadCredentials(
        password, walletfile);

    # defines the operation bid on the artifact #
    @OPERATION public void bid(double bidValue, string agentBidding)
    {
        # instantiates the contract wrapper #
        AuctionMAS auctionContract = AuctionMAS.load("0x0401...",
            web3, credentials, gasPrice, gasLimit);
...
        # makes a bid on the Ethereum Smart Contract #
        TxReceipt txReceipt = auctionContract.placeBid(
            task,
            bidValue,
            agentBidding).get();
...

        # retrieves who is the current winner from the Smart Contract #
        String currentWinner;
        currentWinner = auctionContract.getCurrentWinnerbyAuctionID(
            task).get().getValue();
```

. . .

The code was largely simplified[1] for better comprehension, but it highlights how the artifact of the MAS loads the classes generated by Web3j to make the interface between the agents and the network. The logic of the functions are taken from the artifact and implemented in the Smart Contract, but from the point of view of the agents, nothing changes.

3.3 Conclusion

One of the main challenges from this point on is how to design agents that will reason upon these characteristics provided by blockchains. Can we make agents distinguish the hard consequences of recording data in the blockchain rather than another database that can more easily be tampered with? This approach creates new possibilities for the advancement of Multi-Agent Systems.

The conclusion of this chapter is that blockchains bring some disadvantages that must be taken into account when designing a MAS that will be integrated with this technology. There is a great trade-off between performance and costs for the advantages brought by the Blockchain (security, reliability, flexibility, etc.). The next Chapter exemplifies this case, where some functions required by agents are modeled into artifacts that do not connect to the blockchain, whereas the critical feature (transaction of assets) is done in a Smart Contract embedded into an artifact. In applications where there are loose requirements of speed and costs, then it is possible to design every needed interaction between agents through the blockchain. This chapter highlights Blockchain's limitations in order to provide a realistic scenario for the application of this technology to Multi Agent Systems, but it is important to note that these restrictions are imposed by the current state of the technology, and that in the future it is very possible, if not probable, that improvements are made to both the protocol and real world infrastructure, which could make the usage of Blockchain as easy, cheap and accessible as, say, the Internet today.

[1] full code available at `https://github.com/FerPapi/HouseBuilding_JaCaMo-Blockchain`

4 An application example

Up to this point, this work discussed the technical foundation of Blockchains and Situated Artificial Institutions, and how they might relate regarding the regulation of transaction of assets. Chapter 3 proposes the framework for the design of MAS with Blockchain, where different Smart Contracts are embedded in different artifacts, and that not necessarily every artifact needs to be implemented via Smart Contracts. To illustrate these concepts, this chapter proposes an extension to a problem that is commonly used to illustrate MAS concepts: the decentralized construction of a house by different competing agents. This example, which will be called *build-a-house*, is given in [8], extending the original implementation given in [6]. This is done in three steps: First, the original problem is extended to include a step where agents are paid to do their job. This payment is done exclusively inside the definition of the MAS, and there is no regulation of this system via SAI. Second, the implementation provided by [8] is also extended to include a different mechanism of payments, which are regulated by the Institution. Third, the blockchain is integrated to the system, that is, the MAS has a SAI based regulation and artifacts that support transaction of assets via blockchain. This way, it is possible to compare the problems that arise when implementing solutions that are more simple and naïve, and highlight the solutions and benefits brought by the proposed technologies.

4.1 Original problem with a simple payment extension

The objective of this section is to introduce the problem being proposed coupled with a simple payment layer. This example lays the ground for the added complexity in the next sections. The *build-a-house* example is a MAS that simulates the "construction" of a digital house following an inter-organizational workflow. In said MAS, there is an agent with the role *House Owner* called *Giacomo* whose goal is to have a house built for himself. This agent will hire other autonomous agents, called *contractors* or *companies*, by an auction. *Giacomo* wants to pay as little as possible for this house, so he will launch auctions to determine which *contractor* will do each step of the construction (prepare the site, lay floors, build walls, build the roof, fit windows, fit doors, install plumbing, install the electrical system, paint the exterior, paint the interior). Only when all the steps are performed, the house is said to be built and the goal is completed. Each *company* will participate in the auctions that they see fit, based on their ability to complete the proposed task and how much they will earn. Once there are bids for each auction, the lowest one wins when *Giacomo* closes the auction, meaning he will pay the least amount possible for each auction. The companies are then hired for the jobs and proceed to perform what they were contracted for. Their work must be coordinated, because tasks follow a logical order. For example, the ground must be ready for the floor to be laid, the floor must be ready for the construction of walls, which in turn precedes the construction of the roof, and so on. This coordination is provided by a $\mathcal{M}$oise organization, where agents will play roles that make them responsible for goals regarding the building of the house.

The full diagram containing the goals, sub-goals and their logical flow is presented on Figure 12. A goal or sub-goal is considered complete when all of its sub-goals are completed. In the case of this MAS, all sub-goals have the logical operator *AND*, meaning all sub-goals are mandatory. For example, to have the goal *Build Home* completed, the system needs to have the sub-goal *Install Structure* completed, which in turn requires sub-goals *Plumbing* and *Electric*. For the sake of visualization, the diagram shows sub-goals {*Contract(τ)*, *Construct(τ)*, *Pay(τ)*} grouped into one sub-goal named *CCP(τ)* for a particular goal τ. Sub-goals can have a logical precedence, noted by dashed arrows in the diagram. When there isn't a logical precedence, sub-goals are said to be parallel, and can be executed in any order. For example, sub-goal *Build Wall* precedes *Complete Structure* and must be completed first, but the sub-goals {*Build Roof*, *Install Windows*, *Install Doors*} are executed in no particular order or even concurrently.

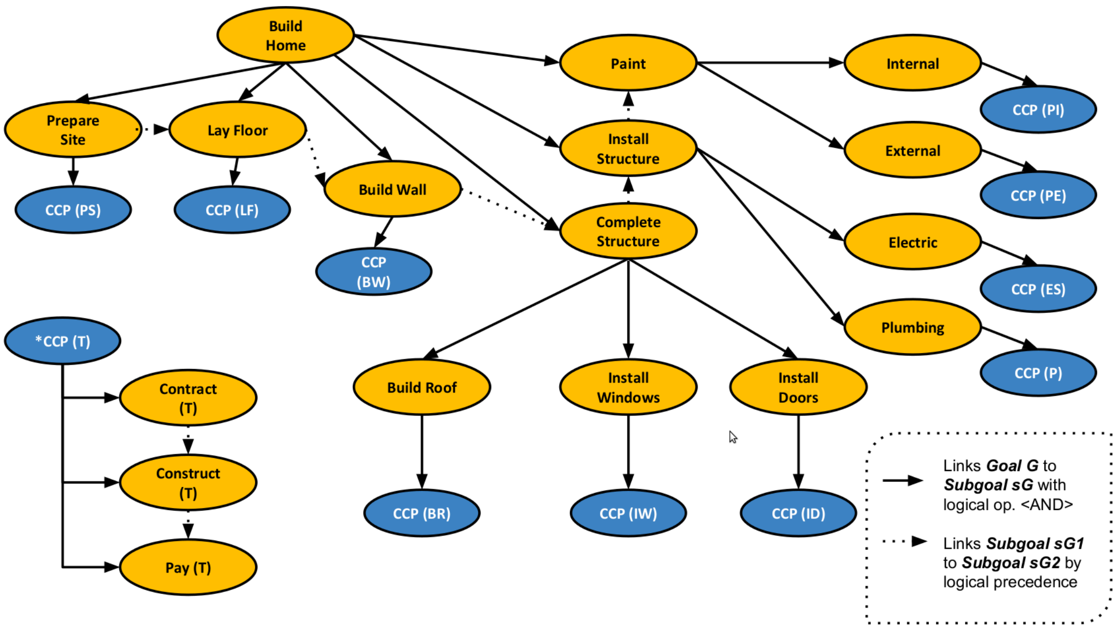


Figure 12 – Goals diagram for the *build-a-house* example

These goals are assigned to specific *Agent Roles*. Agent Roles in this MAS are: *home_owner* and *building_company*. Agents acting as *building_company* have an extension to their role (*bricklayer*, *site_prep_contractor*, *roofer*, *window_fitter*, *door_fitter*, *plumber*, *electrician*, *painter*) to further specify which goals can be assigned to them. Figure 13 shows the assignment of goals to Agent Roles

In this particular implementation, agent *Giacomo* will proactively assume the role of *home_owner*. In order to complete his goals, *Giacomo* needs to contract companies to complete the construction by launching *Auction Artifacts*. These are specified in Figure 14. The Auction Artifact specifies which task it was created for (*task*), what is the maximum value that Giacomo will be willing to pay for the task (*maxValue*), the current winning bid (*currentBid*), and the current winner (*currentWinner*). Whenever a company performs the operation *Bid* on the artifact, it will check if the value proposed by the company is smaller than *currentBid*, and if so, will update *currentWinner* and *currentBid* with the name of the company and the new

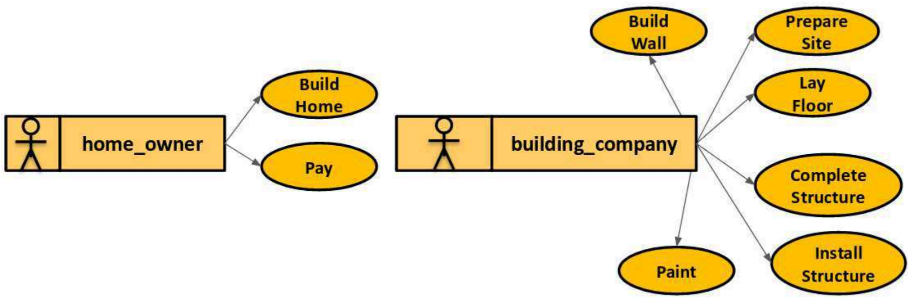


Figure 13 – Assignment of goals to Agent Roles

value, respectively. When Giacomo closes the auction, the company specified in *currentWinner* is expected to adopt the role regarding the task it was contracted for. The actual performance of the task, in this example, is purely illustrative (the system just prints a drawing simulating the respective task) and contributes little to this work, but is shown in Figure 15.

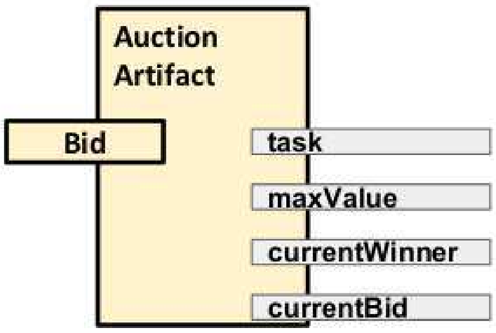


Figure 14 – Auction Artifact

Once every step of construction is done, Giacomo needs to pay the respective contractor. At the beginning of the execution, Giacomo has a belief of some initial balance, e.g,

```
balance(1200);
```

When a company finishes a task, Giacomo will simply subtract the agreed value from his *balance* belief and tell the company that it was paid. The company will just add that amount to its *balance* belief, believing it just got richer:

```
## Giacomo agent code##
+!pay_for_task(Task,GroupName) : task_roles(Task,Roles) & balance(B)
   <-
   ?task(Task)[artifact_id(ArtId)];
   ?currentWinner(Ag)[artifact_id(ArtId)];
   ?currentBid(Price)[artifact_id(ArtId)];

   # discounts the price it is paying from
   #  belief base #
   -+balance(B-Price);

   # tell the contracted agent it was paid #
   .send(Ag,tell,service_paid(Me, Price)).
...

## Company agent code
+service_paid(House_Owner, Price)[source(House_Owner)]: balance(B)
    <-
    # adds the value paid to the balance belief #
    -+balance(B+Price).
```

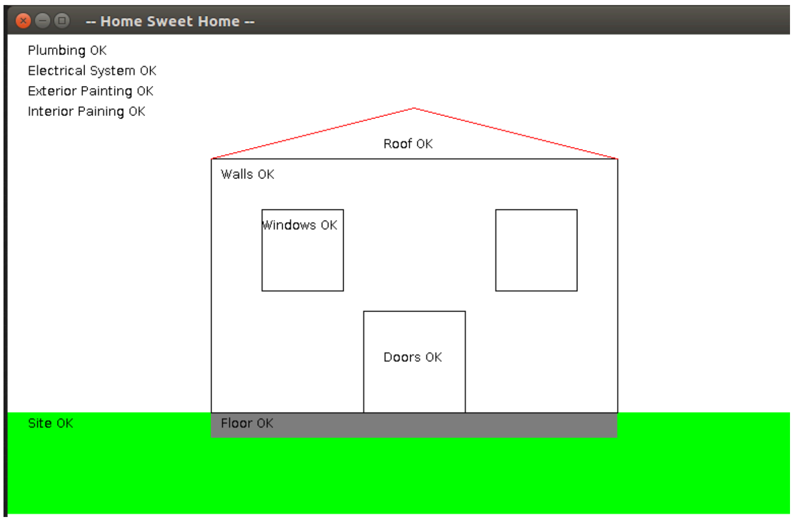


Figure 15 – The outcome of the MAS, a digital house built for Giacomo

This is, of course, a very naïve way of introducing payments. First, the asset, which will be called *money* on a very loose sense from now on, **just exists in the beliefs of the agents**. Giacomo could be designed to have any value as his initial balance. He could also be designed to be malicious and not discount the agreed value from his belief, thus creating a double-spending problem from the point of view of other agents. Since contractor companies are basically being told by Giacomo how much money they should add to their balance, they need to trust him that that value is correct. From this naïve implementation, it is clear that there is the need to make that abstract money count as something real or useful later, and some entity to regulate this transaction in order to try to solve those problems. It is appropriate that this notion of assets is not represented only in the minds of the agents, but that it has some representation that goes beyond that of what agents believe.

4.2 SAI Regulation of the problem

In the original implementation, agents are the ones responsible for interpreting environmental facts, proactively informing that they are adopting roles, achieving goals, etc. There is not a connection between whatever happens on the environment (agents performing functions on artifacts) and the normative state. The system is said not to be *situated.* [8] extends the original implementation, provided in [6], by creating a Situated Artificial Institution to regulate the behaviour of those agents.

Recapitulating the goal of this work, there is the need to build an infrastructure for this MAS regarding the transaction of assets, which in this case will be some internal variable loosely representing money. The SAI will introduce *situational based* regulation on this system. By using the *count as* operator, the institution assigns Status Functions to agents and actions regarding what they represent on the system, assigning them deontic powers effectively regulating the problem via constitutive rules. For example, on the problem posed on section 4.1, upon the initialization of the system, Giacomo would assign himself the role of *home_owner*. Because he is the *home_owner*, Giacomo would then launch the Auction Artifacts and hire the winning companies for each task. In this case, the SAI will deal with these assignments differently. The institution includes the following status functions:

- *play(A,R,Gr).* This means that an agent A plays role R in group Gr. When the Institution effectively states *play(giacomo, home_owner, "hsh_group")*, it means that Giacomo will now play the role of *home_owner* under any circumstances (since no clauses *when* and *while* were declared). The assignment of the role *site_prep_contractor* (a extension of the *building_company* role) will need to be assigned only to the company that won the auction for the *Prepare Site* auction. Therefore, the constitutive rule *play(Agent, site_prep_contractor, "hsh_group")* is constituted if the property *currentWinner(auction_for_sitePreparation, Agent)* holds true. This process is repeated for all the other Auction Artifacts and their respective winners, who will be assigned to agent roles accordingly, as shown in Figure 16.

Now that the company agents that won the auctions are assigned to their respective roles, they will need to produce events that count as achieving the goals specified for the MAS. This is done defining the next three status functions:

- *responsible(Gr,S).* A *responsible* state-status function happens in a constitutive state when state-status functions *play* are also in the constitutive state. This means that the group *Gr* (*hsh_group*) will be responsible for executing actions described on the scheme *S* (*bhsch*), constituting the status function *responsible("hsh_group", "bhsch").*
- *commited(A,M,S)* A constitutive rule will need to make agents commited to missions so that they can pursue the goals in that mission. The status function *commited(A,M,S)* will do just that when the status function *play(A,R,Gr)* is also constituted. For example, in

```
status_functions:
    states: play(A,R,Gr)
constitutive_rules:
    1: count-as play(giacomo,house_owner,"hsh_group").

    2: currentWinner(auction_for_SitePreparation,Agent)
        count-as play(Agent,site_prep_contractor,"hsh_group").

    3: currentWinner(auction_for_Floors,Agent)
        count-as play(Agent,bricklayer,"hsh_group").

    4: currentWinner(auction_for_Walls,Agent)
        count-as play(Agent,bricklayer,"hsh_group")
    ...
```

Figure 16 – Definition of *play(A,R,Gr)* status function and constitutive rules derived from it

this scenario, there will be a constitutive rule that states that an agent playing the role *home_owner* is commited to the mission *management_of_house_building* and will pursue the goals contained in that mission.

- *achieved(S,G,A)*. This status function will be used to effectively state that a goal G was completed by an agent *A* when an specified event happens on the Environment. For example, a constitutive rule will define that the status function *achieved("bhsch", roof_built, Agent)* is constituted when *buildRoof[sai___agent(Agent)]* happens on the environment. This will be done for all goals that need to be fulfilled on the MAS.

Examples of these constitutive rules are given on Figure 17. The constitutive rule 14, for example, states that whatever agent that is playing the role *bricklayer* will be the one commited to the mission *lay_floors* (which contains the goal *floors_laid*). The goal *floors_laid* that composes this mission is defined as *achieved* by constitutive rule 24, when the agent commited to achieving the goal produced the event *lay_floors* on the environment. The full definition of the constitutive rules is available on Appendix A [1].

4.2.1 Introducing Payments

All the work described up until now in this section is available in [8] and that work provides more complete explanations about the SAI designed for this system. At this point, the original system of auctions and construction is successfully regulated by the SAI, and we are now able to add another structure of payments in this system that is able to be institutionalized by the SAI. Since the SAI will need events happening on the environment to regulate transactions that count as payments, there is the need of an artifact that provides those functions to agents. This is a crucial point, because we introduce centralization in the design, a point that will be

[1] The definition of the *responsible* constitutive rules is quite more complex, but contributes little to the understanding of this problem, so it will be omitted for simplicity.

```
status_functions:
    states: play(A,R,G), responsible(G,S),
    committed(A,Mission,S), achieved(S,G,A)

constitutive rules:
    12: play(A,house_owner,"hsh_group")
        count-as committed(A,management_of_house_building,"bhsch")
        while responsible("hsh_group","bhsch").

    13: play(A,site_prep_contractor,"hsh_group")
        count-as committed(A,prepare_site,"bhsch")
        while responsible("hsh_group","bhsch").

    14: play(A,bricklayer,"hsh_group")
        count-as committed(A,lay_floors,"bhsch")
        while responsible("hsh_group","bhsch").
...

    22: count-as achieved("bhsch",site_prepared,Agent)
        when prepareSite[sai__agent(Agent)].

    23: count-as achieved("bhsch",electrical_system_installed,Agent)
        when installElectricalSystem[sai__agent(Agent)].

    24: count-as achieved("bhsch",floors_laid,Agent)
        when layFloors[sai__agent(Agent)].
...
```

Figure 17 – Definition of *commited(A,M,S)* and *achieved(S,G,A)* status functions and constitutive rules derived from them

further explored over the next section. This work proposes the creation of a Bank Artifact that will loosely model a real world bank.

All the agents are modified to include a belief of an initial balance, as done before. When they are instantiated in the MAS, they are required to go to an artifact called *Bank Artifact* and “open an account" with this initial balance. Giacomo starts with some predefined value sufficient for the payments made later on and contractors start with the initial balance of 0. This bank artifact has a hash map, a table of {key: value} pairs that will keep track of agents’ balances. We may call this pair the agent’s *Account* on the bank. Agents are able to perform a number of operations on this artifact:

- *makeAccount*. The bank will create an entry on the hash map with the name of the agent as key, and the initial value declared by the agent as value;

- *depositValue*. The bank will add the specified amount to a client’s Account;

- *checkValue*. The bank will return the value of the Account of the agent making the requisition. The agent can, then, update its belief base with the value provided by the bank, so that he can internally represent this money in his beliefs;
- *transferValue*. An agent can request the transfer of an amount to another agent's account. The value will be subtracted of its account value, and the same value will be added to the recipient account value. The bank will perform a minimal security check on this operation, checking if there is an Account for the recipient of the transaction, and if the requesting Agent has enough funds on his Account to complete the transaction.

Figure 18 illustrates this artifact. There are no observable properties available for this artifact. After introducing the artifact, the SAI must be linked to it in order for its operations effectively take part in the regulation of the system. This is done similarly to the previous constitutive rules, and no new status functions will be needed.

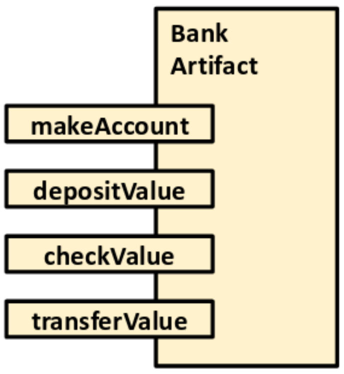


Figure 18 – Bank Artifact

Since Giacomo is already assigned to the role *home_owner*, the SAI will need to state that whatever agent that takes this role needs to commit to the mission *payments*. The mission *payments* is a set of all goals relating to the payment of every step of the construction. Figure 19 shows some examples of constitutive rules that state when a payment goal is achieved. Taking constitutive rule 202 in this example, the goal that states the payment for the construction of the walls, *build_walls_paid*, is considered (*count-as*) achieved when an agent (Giacomo) performs the operation *transferValue(V,X)*, where *V* is the value specified in the observable property *currentBid* from the *auction_for_Walls* artifact, and *X* is the agent playing the role of *bricklayer*, which was defined by the SAI in the constitutive rule 4, shown in Figure 16.

Now this model successfully added a layer of payments for the agents' services and lays the ground for integrating a blockchain to the system. From a strict point of view, the agents have an abstract notion of money in their belief base, which can be considered as an asset, because for these agents that money is valuable for something. Giacomo needs it to pay for the construction of the house, to achieve the goals of payments as defined by the system. The SAI successfully regulates these payments, because Giacomo now has an incentive to transfer some of his money if he wants to achieve the system's goal of paying for the service. One limitation still

```
constitutive_rules:
...
100: play(A,house_owner,"hsh_group")
     count-as committed(A,payments,"bhsch").
...
200: count-as achieved("bhsch",prep_site_paid,Agent)
     when transferValue(V,X)[sai__agent(Agent)]
     while play(X,site_prep_contractor,"hsh_group")
     & currentBid(auction_for_SitePreparation, V).

201: count-as achieved("bhsch",lay_floors_paid,Agent)
     when transferValue(V,X)[sai__agent(Agent)]
     while play(X,bricklayer,"hsh_group")
     & currentBid(auction_for_Floors, V).

202: count-as achieved("bhsch",build_walls_paid,Agent)
     when transferValue(V,X)[sai__agent(Agent)]
     while play(X,bricklayer,"hsh_group")
     & currentBid(auction_for_Walls, V).
...
```

Figure 19 – Definition of constitutive rules that regulate payments in the MAS

present is that this money is still only valuable on the context of this particular MAS. Even from this simple example, it is easy to see that the Bank Artifact adds a layer of trust to the system, because other agents are no longer dependent on Giacomo to tell them how much they own. But agents are still required to trust that the artifact behaves fairly. They have no other option other than trusting this third-party entity to register how much they have of this asset, and they do not know who designed and launched it. They need to trust the Institution that regulates the system, which in turn “decides" which artifact will provide the functions that count as payments. Adding the blockchain as a “door to the outside world" helps solve some of these issues, and this will be discussed over the next Section.

4.3 The Blockchain Integration

In the examples provided so far, agents have an abstract concept of something they utilize as money, something they use to achieve some predetermined goals. Tough this approach might be sufficient for some specific cases, it falls short in actually giving agents a support for transaction of assets. One starting issue is that this register of value will be kept for only as long as this system is executing, since there is no connection to any outside system that works as a database. Persistence of information, in this case, is a crucial feature missing in the previous implementation, and severely limits the use cases of such a MAS. Another problem is that agents are required to trust the Bank Artifact to handle their assets. Agents have no further information of this artifact, and all they know is that they will receive some value for some task they perform.

Even if the whole system was modeled with persistence and reliable level-of-service in mind (agents would be able to access their money at any point in time in the future), this model cannot escape the trap of centralization. The server or servers where it runs are subject to attacks and the database entries could be modified by a malicious agent. The assets that the agents have put effort into earning could be rendered entirely worthless.

Another issue is that the money that Giacomo uses to pay for the construction of the house means something only in the context of this particular MAS, because it is in this context that the SAI makes it count as payment. If those agents wanted to spend the balance they have on the bank artifact, they would need to spend it in the context of another MAS where a similar Institution would need to make it count as payment for other goods and services.

These issues can be solved by the introduction of the Blockchain as the means of payment in the system. Retrieving the concepts introduced in Section 2.3, the blockchain will act as a persistent and fraud-proof database for transactions of these values. It will introduce the mechanism of trust-less consensus and it can be a bridge between the MAS and other systems regarding the value earned by the agents in exchange for their services. The general approach is to “embed" a Smart Contract in an artifact.

As explained in Chapter 3, the blockchain will be integrated to one artifact of the environment. That means that there will be a Smart Contract with the same functions available in the bank artifact, and it will behave the same. The *Bank Artifact* will now become an interface that connects the agents and the Institution in this MAS to the functions available in the Smart Contract. Giacomo will still interact with this artifact to transfer values to the accounts of the contractors, generating events on the environment that the SAI will use to make it *count as* a payment. The actual implementation of the agents and the SAI does not need to change, but the Bank Artifact will need to be designed to interact with the Smart Contract.

4.3.1 The Smart Contract and The Interface Artifact

The Smart Contract is a programmable code that will be compiled and deployed to the Ethereum Network, in order to interact with Ethereum's blockchain, as described in Chapters 2 and 3. In this problem, the Smart Contract will be implemented in a way that is as similar as possible to the Bank Artifact described previously. It also contains an equivalent of a hash map that records {key: value} pairs to keep track of transactions and agents' balances. Agents are able to make deposits in their accounts, transfer values to other agents and consult their own balance to update their belief base. Figure 20 presents a snippet of the Smart Contract code, and the full code is presented on Appendix B.

When the Smart Contract code is compiled, it generates a bytecode that the Web3j library is able to read and generate Java classes that can communicate, via JSON-RPC, with the code that was deployed to the network. These classes are then imported in the interface artifact, as explained in Section and shown in details in Figure 11.

```
contract BankArtifactMAS {

    # defines what is an Account #
    struct Account {
        bool accountActive;
        string client;
        uint256 accountValue;
    }

    # creates a hash map {clientName: Account}
    mapping (string => Account) bank;

    function CreateAccount( string memory _clientName,
                            uint256 _initialValue) public {
        # if not active, create a new account with the client name
        # and set initial value #
        Account storage newAccount = bank[_clientName];
        if (!newAccount.accountActive) {
            newAccount.client = _clientName;
            newAccount.accountValue = _initialValue;
            newAccount.accountActive = true;
        }
    }

    function checkValue(string memory _clientName) public view returns
     (uint accountValue) {

        # retrieve account and retur value #
        Account storage account = bank[_clientName];
        return account.accountValue;
    }

...
```

Figure 20 – Excerpt of the Bank Smart Contract

This artifact will be responsible for providing the interface with the operations on the Smart Contract. In this implementation, it will also be responsible for managing the local Ethereum account, which in turn is the effective owner of the Smart Contract (from the point of view of the Ethereum Network). Figure 21 shows a simplified schematic of this interfacing between Ethereum's network and the MAS, similar to Figure 11.

Now the agents can act on this interface to manage their assets. For example, Giacomo needs to pay for the services he hired, so he will act on this interface, which will make the necessary requests on the network. Since the artifact manages the local Ethereum account, knows its password and knows the address of the Smart Contract in the blockchain, it will be able to read the state of the Contract when it restarts if the system stops being executed. The persistence and reliability of this information is guaranteed, as explained on Chapter 2. All the MAS needs

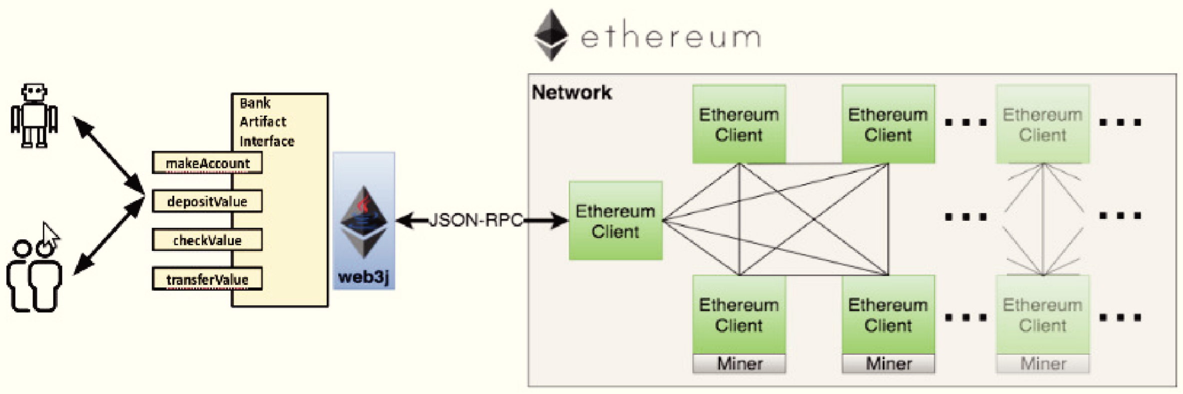


Figure 21 – Bank Artifact Interfacing Ethereum's network

is access to an Ethereum node synchronized to the network, which can be executed locally or remotely.

On one execution of this system, the code was deployed to the address

`0xFd9ACF15Dea1a1ec63bD04F7Bc9B0CAC2b407441`

of Ethereum's test network called Kovan. The transactions made on the contract can be verified by checking the web address given by the QR Code in Figure 22.

Figure 22 – QR Code for https://kovan.etherscan.io/address/0xfd9acf15dea . . .

All that is left is that this balance on the bank still represents some made-up value by the agents (this is specific for this implementation, because agents could have used, from the beginning, their Ether balance in the Smart Contract to deal in the MAS). To solve this, the contract could have some representation of an exchange rate from this money to Ether, and then handle the transactions of Ether from the contract to Ethereum accounts controlled by the contractors. The contractors, then, could take this amount and spend it on whatever they want to spend, convert it to another currency, and so on.

Since this Bank Artifact Interface provides functions for the agents to interact with, this functions will produce events that will be used in the definitions of constitutive rules by the SAI. The system, in this example, is now complete: agents are able to use a real form of money (a currency that they can transport and use in another system in a totally different context).

4.3.2 The flexibility of Smart Contracts

The objective of this work was to propose a model suitable for the transaction of different classes of assets in MAS, and not only something that is regarded as money. To understand why this was in fact achieved, we can imagine a situation where Giacomo, now the owner of a Digital House, engages in another MAS that is actually a combat game. In this game, agents gain some advantage for owning a house (say, if Giacomo took damage for engaging in combat with some enemy, he would be able to regenerate life points of some sort). To prove they have a house, the agents in this game must have a picture (such as the one provided in Figure 15), some unique identifier (like a password for access that Giacomo invents for this house) and his personal Ethereum address. He could then proceed to register this information in a suitable Smart Contract provided by the game, registering the hash of the picture, the password and his Ethereum account. If this system was regulated by a SAI, Giacomo would *count as* a *home_owner*, being allowed to recover life points. Giacomo could want to exchange this house for some other asset, like a weapon, which would also have a specific registering method in the Smart Contract. By sending this other agent the picture (so that the agent can hash its information and verify its veracity on the Contract) and the password, this other agent would be able to change the password for something only he knows, and his own Ethereum address would be recorded as the new owner of this house. Giacomo would do the same with the weapon he just “received", becoming the owner of the weapon and being able to cause more damage to enemies. These agents engaged in the exchange of assets without involving money, be it some internal value like the one described in the previous examples, or even Ether. Giacomo could, for example, sell whatever assets he has on this game for Ether and move on to another system with this money in his account, thus making his earnings in different Systems durable and portable.

4.4 Conclusion

This chapter has shown an illustrative example of the concepts explored mainly in Chapter 2 and 3. By taking an incremental approach, it is possible to understand and highlight the main advantages of the Blockchain and the regulation done by the SAI. Though simple, this example shows that the integration of the technologies was indeed viable and it provides insights and directions to build more complex systems.

5 Conclusions

The main contribution of this work is a model for the Blockchain supporting transaction of assets in Multi-Agent Systems regulated by Situated Artificial Institution. This contribution derives from the work developed to answer the questions proposed in Section 1.1.

- **Is it possible to represent the notion of assets?** Yes, but not without some further considerations. This is demonstrated mainly in Section 4.2.1. In Chapter 2.2 it was argued that assets fall into a category of intangible concepts that humans create to better regulate their interactions with the society around them, the so called *Social Reality*. The analogous process of basing rules and norms from these intangible concepts in a society of autonomous agents is called *institutionalization*, given that an entity called Institution will base the rules for this society on the facts that arise from its environmental elements. The Institution is capable of regulating the behaviour of agents even when the abstract concept is an internal belief of some shared monetary value, because these agents are able to interact with an environmental element to make this concept concrete. Chapter 4.2.1 introduces an example where the concept of assets is successfully institutionalized, and the system is regulated by the Situated Artificial Institution. But this model might not be sufficient to cover all the requirements for the transaction of assets, specially digital monetary assets. Monetary assets require some specific properties (covered briefly on Chapter 1 and Section 2.4.2) so that agents can reliably retain the value of these assets over time. These requirements for digital money were achieved by a technology called Blockchain, which was then used in Section 4.3 to illustrate how the integration of this model actually supports the transaction of assets, not only monetary but of other natures, when integrated with a MAS regulated by SAI.

  It can be argued that the institutionalization of assets was done only in an *indirect* way. The way it was presented, the SAI regulates the transactions of assets, or what *counts as* payment, but **not the object of payment itself**. For example, there are no constitutive rules that make Ether (or whatever other currency the system is using) *count as* money or monetary value, and it is up to the agents to accept the exchange. It is an indirect institutionalization because the agents freely and intentionally agree to the exchange of goods (currency for work effort), therefore they must, by definition, see value in what they are receiving. But because value is a completely subjective concept, the answer to this question could actually be **“No"**. On the next Section, which discusses Future Works, this is further explored.

- **Is integrating a Blockchain into a MAS viable? If so, what is the best role or model for a Blockchain in a MAS?**

  Yes, it is, but not without taking some drawbacks into account. This work proposes that the best model for a Blockchain in a MAS is in the form of Smart Contracts running integrated with different *Artifacts*, covering only critical actions that need integrity of registers, reliable access to critical information, modeling of assets and monetary values, promises, contracts and so on. The answer to this question is mainly covered in Chapter 3. That Chapter proposes a model for the integration of these technologies, taking into account Blockchain's advantages and disadvantages. It was argued over Chapters 1 and 2.3 the properties of Blockchains that make it suitable as digital money, reserve of value and digital transactions system. The main advantage of this technology, and the reason why it was created, is the capability of operating as digital money without the requirement of trust in a centralized third party.

Section 3.2.1 clarifies the trade-off for the advantages brought by Blockchains. This technology is very slow in terms of computing speed, it could become quite expensive to have complex code running on it for large systems and scalability is a serious threat to the mass adoption of Blockchains.

- **Is it possible to model and simulate a problem or system with MAS + Blockchain and what are the advantages and disadvantages of this approach?**

  Yes, it is. An illustrative example is given on Chapter 4, more specifically on Section 4.3. That chapter starts by laying out a simple problem where agents need to cooperate in order to complete the distributed task of building a Digital House. To illustrate the concepts explored in this work, mechanisms of payments and regulation were introduced with increasing complexity. First, agents perform the tasks they were hired for and are "notified" of a payment by the hiring agent. Therefore, they acquire a notion of asset or monetary value, in a very naïve sense. Because of the simplicity of this solution, agents are required not only to trust the system that they will receive something for their work, but they are also required to trust that the paying agent behaves fairly.

  On a second implementation, agents have a "bank account" where they will receive these payments, allowing them to remove the necessity of trust on the paying agent, but requiring them to trust in the fairness of the artifact they are interacting with. The inclusion of a Situated Artificial Institution makes this intangible concept of monetary value concrete from the point of view of the regulation of the system, and is a sufficient approach when there is no need for the assets to hold their values on the long term. Another issue with this implementation is that the representation of money is valid only in the context of that specific system, and the payoff for the work they have done in the system will very likely not be portable to another system.

  Finally, a Blockchain was integrated to the whole system. Now agents do not need to be concerned with the long-term value of the assets they are receiving, because the blockchain works as a reliable database that will hold the value of their assets for as long as it exists (granted that this value may vary with the price of the currency they are receiving, the Ether, as anything subject to the forces of open markets). The trustworthiness of the financial transaction does not depend on the belief of agents, as it does in the first two implementations. The Smart Contract adds enough flexibility so that a variety of assets can be exchanged using this approach, as long as they can be uniquely represented in a Smart Contract, as explained in Section 4.3.2.

With these answers in mind, the conclusion is that Blockchain is a valuable addition to Multi-Agents Systems as a support for the transaction of assets. If the system is able to loose its requirements of speed and costs of operation, it will have included the features of reliability, decentralization, transparency and flexibility regarding transaction of assets. This can become particularly useful in complex systems that involve commerce and transaction of values, decentralization of information and that are not particularly dependent on speed. Such systems were briefly cited before, but Supply Chains [9] and Small Scale Distributed Smart Grids [50, 47] fit particularly well in these categories. Moreover, the integration with the SAI proves to be a successful way to regulate the behaviour of these agents when interacting in the transaction of assets with the Blockchain. More specifically, recalling Figure 8, it is possible to say that the usage of a Blockchain to the proposed problem has successfully added “Data Integrity”, “Traceability”, “Transparency” and “Authenticity” to the system.

5.1 Future Work

By developing this dissertation, some points have been observed that could be investigated in future work. Some of them are described below:

- **The “direct” institutionalization of assets** When answering the question “Is it possible to represent the notion of assets?” in the previous section, it was argued that the institutionalization of assets could be considered only “indirect”, because the implementation of the SAI only considers what *counts as* payments, and not what *counts as* money. One way to overcome this situation would be to define what the institution *counts as* transfer of funds and define that this transfer of funds *counts as* the achievement of the payments goals, as drafted below:

```
1: transfer_value(Value,Ceditor)[sai__agent(Debtor)] count-as
fund_transfer(Debtor,Creditor,Value)

2: count-as achieved("bhsch",prep_site_paid,Debtor)
when fund_transfer(V,Debtor,Creditor)
while play(Creditor,site_prep_contractor,"hsh_group")
& currentBid(auction_for_SitePreparation, V).
```

  In its current state, the SAI does not support the institutionalization of the concept of money. Status Functions are only applicable to agents, states and events, and “money" does not fall into any of these categories. This could also be the subject of study of further works.

- **Improvements on the Blockchain integration** During the implementation of the illustrative example, some simplifications were made and some shortcuts were taken. For example, the Bank Artifact Interface handles a single local Ethereum account, and agents use this account (via the artifact) to interact with the Blockchain. A more reasonable approach would be that agents have their own Ethereum accounts, which they register on the Smart Contract and make some funds available (and retrievable) from this Contract. A future work could provide a framework or guideline about how these design decisions impact the overall performance, robustness and security of the system.

- **A small scale example from the real world** The illustrative examples shown in Chapter 4 creates a starting point for this integration of Blockchains and MAS. It would be valuable to have some real world systems modeled and simulated in MAS utilizing the Blockchain to observe if there are further gains or advantages when using the Blockchain technology. Supply Chains and Distributed Energy Systems, again, are two good examples.

- **Porting the Institution to the Blockchain** Having the Institution modeled and operating on the Blockchain would mean that it would also become reliable, immutable and transparent. It could act as a sort of “lifetime" judge, enforcing rules and attributing roles to agents, making sure goals are completed, etc.

- **A specific blockchain for MAS** Ethereum is reliable, flexible and sufficiently user-friendly, with a large community of engineers, developers and users. But it is also a large, public, “general-purpose" Blockchain, with worse than desired speed and costs. A lighter and more specific blockchain supporting common features for agents could be researched and proposed.

Bibliography

[1] Andreas M Antonopoulos. *Mastering Bitcoin: unlocking digital cryptocurrencies.* " O'Reilly Media, Inc.", 2014.

[2] Alexander Artikis and Marek Sergot. Executable specification of open multi-agent systems. *Logic Journal of IGPL*, page jzp071, 2009.

[3] British Blockchain Association. British blockchain association main website, 2019. Available at *https://www.britishblockchainassociation.org/jbba.*

[4] Bitcoin and Cryptocurrency Technologies Online Course Ed Felten. Lecture 1 — intro to crypto and cryptocurrencies.

[5] Guido Boella, Leendert Van Der Torre, and Harko Verhagen. Introduction to the special issue on normative multiagent systems. *Autonomous Agents and Multi-Agent Systems*, 17(1):1–10, 2008.

[6] Olivier Boissier, Rafael H Bordini, Jomi F Hübner, Alessandro Ricci, and Andrea Santi. Multi-agent oriented programming with jacamo. *Science of Computer Programming*, 78(6):747–761, 2013.

[7] Rafael H Bordini and Jomi F Hübner. Bdi agent programming in agentspeak using jason. In *International Workshop on Computational Logic in Multi-Agent Systems*, pages 143–164. Springer, 2005.

[8] Maiquel de Brito et al. A model of institucional reality supporting the regulation in artificial institutions, 2016.

[9] Paul Brody. How blockchain is revolutionizing supply chain management, 2017. Available at *https://www.ey.com/Publication/vwLUAssets/ey-blockchain-and-the-supply-chain-three/$FILE/ey-blockchain-and-the-supply-chain-three.pdf.*

[10] K. Brousmichc, A. Anoaica, O. Dib, T. Abdellatif, and G. Deleuze. Blockchain energy market place evaluation: An agent-based approach. In *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 321–327, Nov 2018.

[11] Vitalik Buterin. Meta: cap total ether supply at 120 million, 2018. Available at *https://github.com/ethereum/EIPs/issues/960.*

[12] Vitalik et. al Buterin. Ethereum white paper, 2013.

[13] D. Calvaresi, A. Dubovitskaya, D. Retaggi, A. F. Dragoni, and M. Schumacher. Trusted registration, negotiation, and service evaluation in multi-agent systems throughout the blockchain

technology. In *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pages 56–63, Dec 2018.

[14] Davide Calvaresi, Alevtina Dubovitskaya, Jean Paul Calbimonte, Kuldar Taveter, and Michael Schumacher. Multi-agent systems and blockchain: Results from a systematic literature review. In *International Conference on Practical Applications of Agents and Multi-Agent Systems*, pages 110–126. Springer, 2018.

[15] A. Chauhan, O. P. Malviya, M. Verma, and T. S. Mor. Blockchain and scalability. In *2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pages 122–128, July 2018.

[16] Giovanni Ciatto, Stefano Mariani, and Andrea Omicini. Blockchain for trustworthy coordination: A first study with linda and ethereum. In *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pages 696–703. IEEE, 2018.

[17] Sue ES Crawford and Elinor Ostrom. A grammar of institutions. *American Political Science Review*, 89(3):582–600, 1995.

[18] Maiquel De Brito, Jomi Fred Hübner, and Olivier Boissier. Situated artificial institutions: stability, consistency, and flexibility in the regulation of agent societies. *Autonomous Agents and Multi-Agent Systems*, 32(2):219–251, 2018.

[19] Cambridge Online Dictionary. "asset", 2019. Available at *https://dictionary.cambridge.org/dictionary/english/asset*.

[20] Ethereum Foundation. Ethereum official documentation, 2019.

[21] Frontiers. About: Frontiers in blockchain, 2019. Available at *https://www.frontiersin.org/journals/blockchainabout*.

[22] P Froystad and J Holm. Blockchain: powering the internet of value. *EVRY Labs*, 2016.

[23] Jomi Fred Hübner, Jaime Simao Sichman, and Olivier Boissier. A model for the structural, functional, and deontic specification of organizations in multiagent systems. In *Brazilian Symposium on Artificial Intelligence*, pages 118–128. Springer, 2002.

[24] Aleksandr Kapitonov, Ivan Berman, Vitaly Bulatov, Sergey Lonshakov, and Aleksandr Krupenkin. Robonomics based on blockchain as a principle of creating smart factories. In *2018 Fifth International Conference on Internet of Things: Systems, Management and Security*, pages 78–85. IEEE, 2018.

[25] Aleksandr Kapitonov, Ivan Berman, Sergey Lonshakov, and Aleksandr Krupenkin. Blockchain based protocol for economical communication in industry 4.0. In *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, pages 41–44. IEEE, 2018.

[26] Aleksandr Kapitonov, Sergey Lonshakov, Aleksandr Krupenkin, and Ivan Berman. Blockchain-based protocol of autonomous business activity for multi-agent systems consisting of uavs. In *2017 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*, pages 84–89. IEEE, 2017.

[27] Ledger. Ledger journal main website, 2019. Available at *https://ledgerjournal.org/ojs/index.php/ledger.*

[28] Michael Luck, Peter McBurney, Onn Shehory, and Steve Willmott. *Agent technology: computing as interaction (a roadmap for agent based computing).* University of Southampton, 2005.

[29] Sun Microsystems. Rpc: Remote procedure call protocol specification, 1988.

[30] Peter Middleton, Peter Kjeldsen, and Jim Tully. Forecast: The internet of things, worldwide, 2013, 2013. Available at *https://www.gartner.com/doc/2625419/forecast-internet-things-worldwide-.*

[31] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[32] Arvind Narayanan, Joseph Bonneau, Edward Felten, Andrew Miller, and Steven Goldfeder. *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction.* Princeton University Press, 2016.

[33] Greg MP O'Hare, Nicholas R Jennings, and Nick Jennings. *Foundations of distributed artificial intelligence*, volume 9. John Wiley & Sons, 1996.

[34] Andrea Omicini, Alessandro Ricci, and Mirko Viroli. Artifacts in the a&a meta-model for multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 17(3):432–456, 2008.

[35] Steve Omohundro. Autonomous technology and the greater human good. *Journal of Experimental & Theoretical Artificial Intelligence*, 26(3):303–315, 2014.

[36] Fernando G Papi, Jomi Fred Hübner, and Maiquel de Brito. Instrumenting accountability in mas with blockchain. In *CARe-MAS@ PRIMA*, pages 20–34, 2017.

[37] Alessandro Ricci, Michele Piunti, and Mirko Viroli. Environment programming in multi-agent systems: an artifact-based perspective. *Autonomous Agents and Multi-Agent Systems*, 23(2):158–192, 2011.

[38] Alessandro Ricci, Mirko Viroli, and Andrea Omicini. Cartago: A framework for prototyping artifact-based environments in mas. In *International Workshop on Environments for Multi-Agent Systems*, pages 67–86. Springer, 2006.

[39] Stuart J. Russell and Peter Norvig. *Artificial Intelligence - A Modern Approach (3. internat. ed.).* Pearson Education, 2010.

[40] Boris Ryabko and Andrey Fionov. *Basics of contemporary cryptography for IT practitioners*, volume 1. World Scientific, 2005.

[41] John R Searle. *The construction of social reality.* Simon and Schuster, 1995.

[42] George J Stigler. The economics of carl menger. *Journal of Political Economy*, 45(2):229–250, 1937.

[43] Melanie Swan. *Blockchain: Blueprint for a new economy.* " O'Reilly Media, Inc.", 2015.

[44] Michael Szydlo. Merkle tree traversal in log space and time. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 541–554. Springer, 2004.

[45] Friedrich August v Hayek. *Denationalisation of Money: The Argument Refined.* Ludwig von Mises Institute, 1976.

[46] Ludwig Von Mises and Harold Edward Batson. *The theory of money and credit*, volume 2. Yale University Press New Haven, 1953.

[47] Shen Wang, Ahmad F Taha, Jianhui Wang, Karla Kvaternik, and Adam Hahn. Energy crowdsourcing and peer-to-peer energy trading in blockchain-enabled smart grids. *arXiv preprint arXiv:1901.02390*, 2019.

[48] Gerhard Weiss. *Multiagent systems: a modern approach to distributed artificial intelligence.* MIT press, 1999.

[49] Michael Wooldridge. *An introduction to multiagent systems.* John Wiley & Sons, 2009.

[50] Jiani Wu and Nguyen Tran. Application of blockchain technology in sustainable energy systems: An overview. *Sustainability*, 10(9):3067, 2018.

[51] Jesse Yli-Huumo, Deokyoon Ko, Sujin Choi, Sooyong Park, and Kari Smolander. Where is current research on blockchain technology?—a systematic review. *PloS one*, 11(10):e0163477, 2016.

Appendix

APPENDIX A – Constitutive Rules

```
institution_id : bhInst.

status_functions:

states: play(A,R,G), responsible(G,S), committed(A,Mission,S), achieved(S,G,A)

constitutive_rules:

/*In this application, the agent Giacomo is always the house owner.*/
1: count-as play(giacomo,house_owner,"hsh_group").

/*Rules 2 to 10: The state where the property currentWinner(Auction,Agent) holds
counts as Agent playing some role in the house building.*/
2: currentWinner(auction_for_SitePreparation,Agent)
   count-as play(Agent,site_prep_contractor,"hsh_group").

3: currentWinner(auction_for_Floors,Agent)
   count-as play(Agent,bricklayer,"hsh_group").

4: currentWinner(auction_for_Walls,Agent)
   count-as play(Agent,bricklayer,"hsh_group").

5: currentWinner(auction_for_Roof,Agent)
   count-as play(Agent,roofer,"hsh_group").

6: currentWinner(auction_for_WindowsDoors,Agent)
   count-as play(Agent,window_fitter,"hsh_group").

7: currentWinner(auction_for_WindowsDoors,Agent)
   count-as play(Agent,door_fitter,"hsh_group").

8: currentWinner(auction_for_Plumbing,Agent)
   count-as play(Agent,plumber,"hsh_group").

9: currentWinner(auction_for_ElectricalSystem,Agent)
   count-as play(Agent,electrician,"hsh_group").

10: currentWinner(auction_for_Painting,Agent)
```

```
    count-as play(Agent,painter,"hsh_group").


/* Defining what counts as, from the institutional perspective, a well formed group.
This constitution is necessary to constitute the commitments */
11: count-as responsible("hsh_group","bhsch")
    while play(Electrician1,electrician,"hsh_group")&
    not(play(Electrician2,electrician,"hsh_group")&
    not(Electrician1==Electrician2))&
    play(Site_prep_contractor1,site_prep_contractor,"hsh_group")&
    not(play(Site_prep_contractor2,site_prep_contractor,"hsh_group")&
    not(Site_prep_contractor1==Site_prep_contractor2))&
    play(Bricklayer1,bricklayer,"hsh_group")&
    play(Bricklayer2,bricklayer,"hsh_group")&
    not(play(Bricklayer3,bricklayer,"hsh_group")&
    not(Bricklayer1==Bricklayer3)&
    not(Bricklayer2==Bricklayer3))&
    play(Plumber1,plumber,"hsh_group")&
    not(play(Plumber2,plumber,"hsh_group") & not(Plumber1==Plumber2))&
    play(Window_fitter1,window_fitter,"hsh_group")&
    not(play(Window_fitter2,window_fitter,"hsh_group")&
    not(Window_fitter1==Window_fitter2))&
    play(Door_fitter1,door_fitter,"hsh_group")&
    not(play(Door_fitter2,door_fitter,"hsh_group") &
    not(Door_fitter1==Door_fitter2))&
    play(Roofer1,roofer,"hsh_group") &
    not(play(Roofer2,roofer,"hsh_group") &
    not(Roofer1==Roofer2))&
    play(House_owner1,house_owner,"hsh_group") &
    not(play(House_owner2,house_owner,"hsh_group") &
    not(House_owner1==House_owner2)).


/* Rules 13 to 22 (2nd order constitution): An the state of an agent A
carrying the status function Y counts as the state committed(A,M,S)
in the institution */
12: play(A,house_owner,"hsh_group")
    count-as committed(A,management_of_house_building,"bhsch")
    while responsible("hsh_group","bhsch").

13: play(A,site_prep_contractor,"hsh_group")
    count-as committed(A,prepare_site,"bhsch")
    while responsible("hsh_group","bhsch").
```

```
14: play(A,bricklayer,"hsh_group")
    count-as committed(A,lay_floors,"bhsch")
    while responsible("hsh_group","bhsch").

15: play(A,bricklayer,"hsh_group")
    count-as committed(A,build_walls,"bhsch")
    while responsible("hsh_group","bhsch").

16: play(A,roofer,"hsh_group")
    count-as committed(A,build_roof,"bhsch")
    while responsible("hsh_group","bhsch").

17: play(A,window_fitter,"hsh_group")
    count-as committed(A,fit_windows,"bhsch")
    while responsible("hsh_group","bhsch").

18: play(A,door_fitter,"hsh_group")
    count-as committed(A,fit_doors,"bhsch")
    while responsible("hsh_group","bhsch").

19: play(A,plumber,"hsh_group")
    count-as committed(A,install_plumbing,"bhsch")
    while responsible("hsh_group","bhsch").

20: play(A,electrician,"hsh_group")
    count-as committed(A,install_electrical_system,"bhsch")
    while responsible("hsh_group","bhsch").

21: play(A,painter,"hsh_group")
    count-as committed(A,paint_house,"bhsch")
    while responsible("hsh_group","bhsch").

100: play(A,house_owner,"hsh_group")
     count-as committed(A,payments,"bhsch")
     while responsible("hsh_group","bhsch").

/* Rules 23 to 22: the occurrence of some events
in the environment counts-as, in the institution,
the state achieved(S,G,A) */
22: count-as achieved("bhsch",site_prepared,Agent)
    when prepareSite[sai__agent(Agent)].
```

```
23: count-as achieved("bhsch",electrical_system_installed,Agent)
    when installElectricalSystem[sai__agent(Agent)].

24: count-as achieved("bhsch",floors_laid,Agent)
    when layFloors[sai__agent(Agent)].

25: count-as achieved("bhsch",walls_built,Agent)
    when buildWalls[sai__agent(Agent)].

26: count-as achieved("bhsch",roof_built,Agent)
    when buildRoof[sai__agent(Agent)].

27: count-as achieved("bhsch",windows_fitted,Agent)
    when fitWindows[sai__agent(Agent)].

28: count-as achieved("bhsch",doors_fitted,Agent)
    when fitDoors[sai__agent(Agent)].

29: count-as achieved("bhsch",plumbing_installed,Agent)
    when installPlumbing[sai__agent(Agent)].

30: count-as achieved("bhsch",electrical_system_installed,Agent)
    when installElectricalSystem[sai__agent(Agent)].

31: count-as achieved("bhsch",exterior_painted,Agent)
    when paintExterior[sai__agent(Agent)].

32: count-as achieved("bhsch",interior_painted,Agent)
    when paintInterior[sai__agent(Agent)].

200: count-as achieved("bhsch",prep_site_paid,Agent)
     when transferValue(V,X)[sai__agent(Agent)]
     while play(X,site_prep_contractor,"hsh_group")
     & currentBid(auction_for_SitePreparation, V).

201: count-as achieved("bhsch",lay_floors_paid,Agent)
     when transferValue(V,X)[sai__agent(Agent)]
     while play(X,bricklayer,"hsh_group")
     & currentBid(auction_for_Floors, V).

202: count-as achieved("bhsch",build_walls_paid,Agent)
     when transferValue(V,X)[sai__agent(Agent)]
     while play(X,bricklayer,"hsh_group")
     & currentBid(auction_for_Walls, V).

203: count-as achieved("bhsch",roof_paid,Agent)
     when transferValue(V,X)[sai__agent(Agent)]
     while play(X,roofer,"hsh_group")
     & currentBid(auction_for_Roof, V).

204: count-as achieved("bhsch",windows_paid,Agent)
     when transferValue(V,X)[sai__agent(Agent)]
     while play(X,window_fitter,"hsh_group")
     & currentBid(auction_for_WindowsDoors, V).

205: count-as achieved("bhsch",doors_paid,Agent)
     when transferValue(V,X)[sai__agent(Agent)]
     while play(X,door_fitter,"hsh_group")
     & currentBid(auction_for_WindowsDoors, V).

206: count-as achieved("bhsch",plumbing_paid,Agent)
     when transferValue(V,X)[sai__agent(Agent)]
     while play(X,plumber,"hsh_group")
     & currentBid(auction_for_Plumbing, V).

207: count-as achieved("bhsch",electrical_system_paid,Agent)
     when transferValue(V,X)[sai__agent(Agent)]
     while play(X,electrician,"hsh_group")
     & currentBid(auction_for_ElectricalSystem, V).

208: count-as achieved("bhsch",exterior_paint_paid,Agent)
      when transferValue(V,X)[sai__agent(Agent)]
      while play(X,painter,"hsh_group")
      & currentBid(auction_for_Painting, V).

209: count-as achieved("bhsch",interior_paint_paid,Agent)
     when transferValue(V,X)[sai__agent(Agent)]
     while play(X,painter,"hsh_group")
     & currentBid(auction_for_Painting, V).
```

APPENDIX B – Smart Contract Code

```
pragma solidity ^0.5.1;

contract BankArtifactMAS {

    struct Account {
        bool accountActive;
        string client;
        uint256 accountValue;
    }

mapping (string => Account) bank;

constructor() public {
    }

function CreateAccount( string memory _clientName,
                        uint256 _initialValue) public {
    Account storage newAccount = bank[_clientName];
    if (!newAccount.accountActive) {
        newAccount.client = _clientName;
        newAccount.accountValue = _initialValue;
        newAccount.accountActive = true;
        }
    }

function checkValue(string memory _clientName)
  public view returns (uint accountValue) {
    Account storage account = bank[_clientName];
    return account.accountValue;
    }

function depositValue(string memory _clientName, uint256 _value) public {
    Account storage account = bank[_clientName];
    uint256 currentValue = account.accountValue;
    bank[_clientName].accountValue = currentValue + _value;
}

function transferValue(uint256 _transferValue,
                       string memory _clientName,
                       string memory creditor) public returns
                       (string memory confirmationMessage) {
    Account storage sendingAccount = bank[_clientName];
    Account storage receivingAccount = bank[_creditor];
    uint256 sendingAccountValue = sendingAccount.accountValue;
    if (!receivingAccount.accountActive) {
        confirmationMessage = "Error: No account for this client";
        return confirmationMessage;
    }
    if (sendingAccountValue < _transferValue) {
        confirmationMessage = "Error: No funds available";
        return confirmationMessage;
    }
    sendingAccount.accountValue = sendingAccountValue - _transferValue;
    receivingAccount.accountValue = receivingAccount.accountValue + _transferValue;
    confirmationMessage = "Transfer Successful!";
    return confirmationMessage;
}
}
```