Blockchain based peer-to-peer personal finance system: A Token Engineering Approach

Huda Abdirahim

Submitted in accordance with the requirements for the degree of BSc. Computer Science

University of Leicester

Department of Informatics

May 2021

Intellectual Property

The candidate confirms that the work submitted is his/her own and that appropriate credit has been given where reference has been made to the work of others.

Abstract

The rapid growth of technology has lead to the creation of digital platforms that act as a trusted third-party between lenders of borrowers in schemes such as peer-to-peer lending, crowdfunding and rotating and credit savings associations, ROSCA, where members pool together regular savings which are then dispersed to each member in turn. In this scenario, we aim to connect lenders and borrowers in a cost-optimal and secure way. Blockchain technology aims to keep track of a ledger of valid transactions between agents of a virtual economy without the need for a central institution for coordination[29]. These inherent features of blockchain enable a transparent, cost-effective, democratic alternative to existing personal finance models.

Based on this, we propose a blockchain-based model of ROSCA in the context of token engineering, which makes it possible to design and analyse economic systems in a manner consistent with the best practices of systems engineering. Our proposed distributed system compromises three main components: A robust peer-to-peer network over which agents interact, the addition of a rich agent-based model, derived from economics and mathematics, which serves to outline all possible agent strategies allowing us to design incentives and penalties. And finally, mechanisms to facilitate governance of a system without a central governing body.

We validate our model through simulation, intending to examine questions related to how financial rewards, incentives and the number of participants affect the outcome and whether the proposed model is, in fact, fairer, more decentralised and democratic than exiting models. We find the experimental results show the proposed model yields better returns in terms of reward vs cost for participants compared to the centralised model of ROSCA.

Contents

1 Introduction

1.1 Project Aims . . . .

1.2 Project Objectives and Structure of Project . . . .

2 Literature Review

2.1 Overview of personal finance schemes . . . .

2.1.1 Peer-to-peer lending . . . .

2.1.2 Crowd-funding . . . .

2.1.3 Rotating Savings and Credit Associations . . . .

2.2 Blockchain technology . . . .

2.3 Smart contracts . . . .

2.4 Crypto-economics and Token Engineering . . . .

2.5 Discussion . . . .

3 Protocol Design

3.1 Motivation for the proposed model . . . .

3.2 System Mapping . . . .

3.2.1 System Dynamics . . . .

3.2.2 System Requirements . . . .

3.2.3 Assumptions . . . .

3.2.4 Rules and Incentive Design . . . .

3.3 Formalising the design . . . .

3.3.1 Entity relationship diagram . . . .

3.3.2 Stock and Flow Diagram . . . .

3.3.3 Mathematical Specification . . . .
3.4 Modularising the logic and building a model . . . .
3.4.1 Differential Specification . . . .
3.5 Considerations for Implementation of Blockchain . . . .
3.5.1 How can blockchain improve areas of this model? . . . .
3.5.2 Block . . . .
3.5.3 Digital Signature . . . .
3.5.4 Consensus Algorithm . . . .

4 Simulation

4.1 cadCAD . . . .
4.2 Abstraction of peer-to-peer network . . . .
4.2.1 Requirements . . . .
4.2.2 Design . . . .
4.2.3 Validation . . . .
4.3 Agent-based Model . . . .
4.3.1 Requirements . . . .
4.3.2 Design . . . .
4.3.3 Validation . . . .
4.4 Governance . . . .
4.4.1 Requirements . . . .
4.4.2 Design . . . .

5 Simulation Analysis

5.1 What-if Matrix . . . .
5.1.1 Analysis 1 . . . .
5.1.2 Analysis 2 . . . .
5.1.3 Analysis 3 . . . .
5.1.4 Analysis 4 . . . .
5.2 Critical Analysis . . . .

6 On Reflection

6.1 Conclusions . . . .

6.2 Model Improvements . . . . .
6.2.1 Where did the proposed model fall short? . . . . .
6.2.2 Future work . . . . .
6.3 Socio-economic and academic impact . . . . .
6.4 Personal Development . . . . .

7 Bibliography and Citations

List of Figures

2.1 Illustration of Blockchain based crowd-funding. . . . . .

2.2 Functioning of rotating savings and credit associations (ROSCA).[28] . . . . . .

3.1 High-level steps involved in the proposed ROSCA model . . . . . .

3.2 Illustration of System Dynamics . . . . . .

3.3 Illustration of System Rules and Incentives . . . . . .

3.4 Entity Relationship Diagram . . . . . .

3.5 Stock and Flow Diagram . . . . . .

3.6 Differential Specification Diagram . . . . . .

3.7 Layout of Data in a Block . . . . . .

4.1 Agent Default Count . . . . . .

4.2 Collected Amount Compared to Defaulted Amount . . . . . .

4.3 Agents' Payoff Matrix . . . . . .

4.4 Agent Decision . . . . . .

4.5 Agents' Strategies Illustration . . . . . .

4.6 Volume of Types of Activity . . . . . .

4.7 Reward and Cost for Different Agents . . . . . .

4.8 Cumulative Profit for Agents . . . . . .

4.9 Cumulative Contributor Return on Investment . . . . . .

5.1 Monte-Carlo analysis for Honest and Dishonest Volumes . . . . . .

5.2 Monte-Carlo analysis illustrating agent Reward and Cost Volumes . . . . . .

5.3 Simulation with parameters duration and participants at 5 . . . . . .

5.4 Simulation with parameters duration and participants at 8 . . . . . .

5.5 Simulation with parameters duration and participants at 12 . . . . . .
5.6 Simulation with parameter max amount at 200 . . . . . .
5.7 Simulation with parameter max amount at 400 . . . . . .
5.8 Simulation with parameter max amount at 600 . . . . . .
5.9 Variations of Maximum Amount and Dishonest Volume . . . . . .

List of Tables

4.1 Peer to Peer Network Simulation Results . . . . . .

4.2 Agent Based Model Simulation Results . . . . . .

5.1 What-if matrix . . . . . .

Chapter 1

Introduction

The rapid expansion of the internet has created an entirely new ecosystem of **digital finance schemes**, from lending and borrowing on for-profit platforms (peer-to-peer lending) to rotating savings and credit associations (ROSCA) where members pool together regular savings which are then dispersed to each member in turn. These systems rely on a centralised platform or a financial institution to act as an intermediary in facilitating electronic payments and operating other aspects of the scheme.

While this system works well enough, the involvement of a trusted third party incurs an additional cost. Both in terms of a significant transactional cost charged by the platform for matching individuals, and the cost of the platforms' autonomy to make decisions. This leads to biases in terms of who can and can not borrow, a lack of transparency between borrowers and recipients and unrestricted access to participants information. This traditional model isn't as fair and democratised as it initially set out to be.

What is needed is a smart framework that allows two or more parties to lend and borrow or to form a savings circle without the need of a trusted third party. This framework has to be robust enough to handle both the need for governance and data-driven decision making.

In this paper, we propose a model solution of ROSCA that leverages the transparency, traceability and security inherent to distributed ledger technology, to improve the traditional model of ROSCA. In our proposed model, we utilise smart contract technology which has predefined conditions, dictating how funds will flow and executing transactions between parties, which serves to prevent disputes. This is built on top of the blockchain platform which provides the

infrastructure to decentralise the network and facilitate a means of trust-less transactions. In combining the architecture of blockchain technologies with a tokenised incentive system: we propose a single distributed system of ROSCA.

1.1 Project Aims

The primary aim of this project is to address problems with existing personal finance models and develop a design of a blockchain-based system that satisfies requirements such as being fair, transparent and democratic. We then verify our design: rules, mechanisms and incentives in the protocol trigger correct behaviour, through simulation using the token engineering process[19].

Therefore, the main question we will ask and in turn use to evaluate our design and stimulation is two-fold. First, how well does our system incentivise agents, whom we do not control, to self-organise in a way that fulfils the purpose of our protocol? And secondly, is my proposed model more fair, democratic, decentralised than existing systems?

1.2 Project Objectives and Structure of Project

We structure our project, in line with the Token Engineering process (2.4) outlined below.

Ecosystem Definition – *What do we want to build?*

1. The analysis of existing schemes - their problems, degree to which they are centralised/peer-to-peer/transparent.
2. Challenge the concept of using blockchain as a solution – does blockchain improve the current system and make sense to use as a solution. If so, which features of blockchain could I utilise to meet my aim.

Our ecosystem definition allows the definition of robust requirements for our system:

Protocol Design – *How do we build?*

1. Define system goals and properties - outline requirements the system needs to hold up to
2. Formalise design of the distributed system, along with rules, mechanisms and incentives

With our design plan, we can implement and simulate our protocol:

Model Simulation – ***How can we write code and build architecture to enable these systems?***

1. Use of cadCAD to build our three-tiered distributed system

Simulation Analysis – ***What is my system worth?***

1. Overall, does my proposed system satisfy our requirements of a fair, democratic, decentralised, transparent ROSCA system?
2. What are the measurable functions and criteria which allows me to make a correct judgement?

Chapter 2

Literature Review

2.1 Overview of personal finance schemes

In the following, we present the key results of our literature review. First, the three main systems of personal finance. For each system, we will outline key pieces of literature surrounding these models, whilst making use of case studies to illustrate how these models work and potentially identify problems.

Second, the literature on the technology and the method of engineering which forms the basis of my proposed model. And finally, a discussion on our findings and the relation between blockchain and personal finance systems will be presented.

2.1.1 Peer-to-peer lending

Peer-to-peer lending (P2P) – refers to lending and borrowing between individuals through a for-profit online platform, without the intermediation of a traditional financial institution.[1] The original premise of online P2P lending platforms was simple and democratic: a person who wishes to borrow but may not have access or cannot afford high interest rates on traditional loans can be matched to a person who wishes to lend and receive interest far higher than that they would get in a savings account[3].

However, platforms running P2P schemes have somewhat steered from this original premise and have several problems voiced from platform users with transparency being at the forefront[2].

Crosman, in her article 'Can blockchain technology revive peer-to-peer lending?' discusses the

possible revival of P2P lending platforms through the use of blockchain or distributed ledger technology[3]. The author argues “blockchain technology makes it possible to transfer ownership of an asset from one person to another without the need for an intermediary.” This is achieved through smart contracts which don’t require a third party for their execution. She concludes, “this is a very appealing low-cost, high-trust platform that you can build that you couldn’t build before”.

This is further supported by Gonzales in her article ”Blockchain, herding and trust in peer-to-peer lending”[1] who summaries: the gradual implementation of blockchain technology to P2P platforms will ”facilitate safer, transparent and quick access to funds without having to deal with the more complex and costly processes of banks.”

To demonstrate the validity of the ideas voiced in the two articles above, below is a summary of companies utilising blockchain to build a decentralised P2P lending platform.

Case Study

- **Celsius**

  Celsius is building a person-to-person lending platform on a blockchain for people who are holding digital assets, such as bitcoin long-term. Through the Celsius platform, such cryptocurrency investors will be able to earn interest on their holdings while Celsius uses them as collateral for consumer loans[32].

- **Bloom**

  Bloom is developing an identity protocol and credit score that runs on a blockchain. Using AI to assess a person’s creditworthiness over-time[33].

- **Salt Lending**

  Salt Lending has built a mechanism for collateralised lending based on the value being stored in a smart contract on a blockchain. In blockchain-based lending, the identity verification and credit scoring process banks go through today could be replaced with a “distributed reputation” system and a “distributed identity” system[34].

2.1.2 Crowd-funding

Crowdfunding provides businesses with an alternative approach to finding financial support and a way to gain feedback or validation for the products and services they provide [4][5][6].

Crowdfunding via the internet features low barriers to entry, low cost, and high speed, and thus encourages innovation.[7]

In the article 'Tracking the Digital Evolution of Entrepreneurial Finance', the authors suggest "The initial hope was that crowdfunding would democratise access to financial capital"[8], however, this has been the case as many of the incumbent sources of capital are controlled and monitored by a relatively small group of individuals or institutions. This coupled with an increasing lack of transparency between donors, recipients of these funds, and the goals of the program.

Crowdfunding, with the decentralized nature of founders, projects, and financiers, offers a unique opportunity for blockchain to deliver benefits to a broad set of users. Leverage the transparency and traceability inherent to the blockchain model to deliver value to the crowdfunding model.[8]

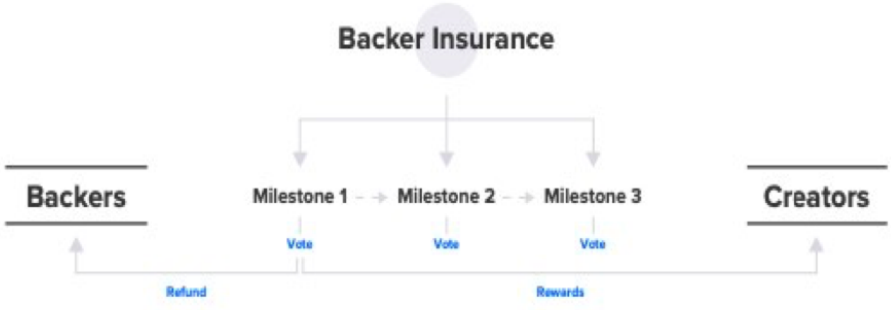

Figure 2.1: Illustration of Blockchain based crowd-funding.

Case Study

- **Pledge-Camp: A blockchain-based crowdfunding platform [9]**
  The foundation of the platform is a smart contract, which both parties enter, defining predefined milestones in which a backer contributes funds which can only be released to the creator through a democratic vote according to whether the milestones are met.

2.1.3 Rotating Savings and Credit Associations

ROSCAs are informal saving and credit institutions that are pervasive in developing countries and immigrant communities. In a ROSCAs, members pool together regular savings with are then dispersed to a member, either by random or based on the greatest need.[10] This continues until every member has received the pot.

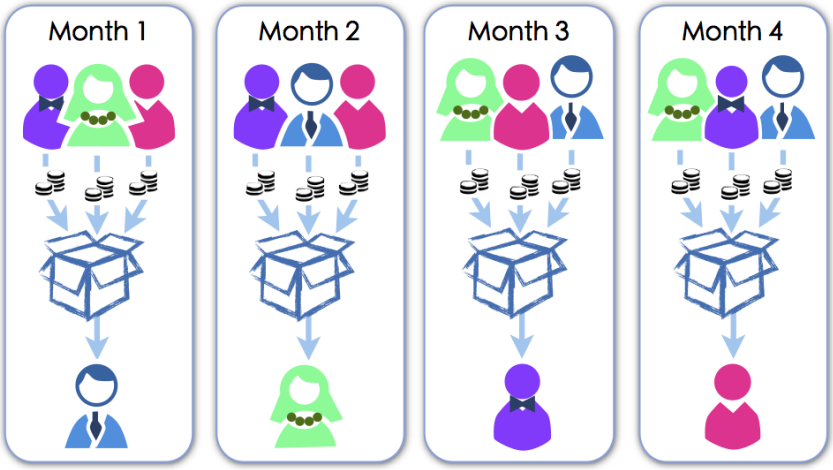

Figure 2.2: Functioning of rotating savings and credit associations (ROSCA).[28]

Why ROSCAs instead of bank loans or perhaps P2P lending? For many, the prospect of running out of money to meet financial obligations is very real. Complicating this problem further is, in a situation like this many live in places where access to credit is poor or doesn't exist[11]. The main distinction of ROSCAs is that many systems are community-based, truly peer-to-peer, without any intermediary and not for profit.

ROSCAs stand to be enhanced massively by "applying a smart contract which automates the savings and lending process. This encompasses contributions, bidding, assigning funds at the end of each round, and withdrawing funds"[11]. This can enable people from across the world to have access to more advanced financial tools.

Case Study

- **Trusted Lending Circles – WeTrust (In development)**

  TLC utilizes the Ethereum blockchain to create a full-stack alternative financial system- this allows for lower fees, improved incentive structures, decentralized risks, allowing a greater amount of capital to reside among the participants, and ultimately improving financial inclusion on a global scale[18].

2.2 Blockchain technology

A blockchain can be described as a decentralised, distributed, peer-to-peer public ledger that is used to record transactions across many computers (so as not to alter records)[12], which was originally conceived as the basis of cryptocurrencies.

From a technical standpoint, a blockchain consists of cryptographically secured blocks that contain transactional data, each chained to the previous by referring to its hash value. The block is then stored on a distributed, peer-to-peer network by a decentralised consensus mechanism (how the network reaches an agreement about which transactions are most trustworthy).

To understand its potential, it is important to distinguish two core blockchain components: the distributed-ledger technology (DLT) described above, and smart contracts.

2.3 Smart contracts

A smart contract - constitutes the rules that participants have collectively agreed upon to govern the evolution of "facts" in the distributed ledger.[12] Such smart contracts can be computer programs that attempt to ensure that all transactions comply with the underlying legal agreements and that the records managed by DLT are reflective of the underlying legal agreements they represent.[15]

2.4 Crypto-economics and Token Engineering

Crypto-economics is an emerging field of economic coordination games in cryptographically secured peer-to-peer networks[19]. It is the combination of cryptography and economics to create a robust decentralised economy in which a collection of economic agents coordinate through peer-to-peer networks of computers via a blockchain protocol.

As we build decentralised economies on blockchain networks, it is important to ensure these complex systems are designed and analysed consistent with modern system engineering. Token engineering is a methodology that goes all the way from ideation to design, modelling, simulation, testing, deployment, and maintenance. It is the process of building crypto-economic systems that work.[16]

2.5 Discussion

The first blockchain system that was originally put in operation was bitcoin, as a means of peer-to-peer payment systems independent of government, banks, or other centralised institutions.[12] From its conception in 2008, the applications of this technology have spread far: from education, governance to personal finance systems.

The main reason for the conception of peer-to-peer platforms and banks, to begin with, is over time we started adding third-party intermediaries because we stopped trusting each other. Glaser[13] argues that blockchains "could provide the infrastructure to decentralise intermediary services and means of trust-less payments". In addition to this, a recent discussion by IBM suggests that blockchains have the potential to "overcome trust-related issues and hence contribute to the resolution of one of the fundamental challenges of peer-to-peer markets".[14]

Going back to our research question of 'Can we build a model solution of personal finance that is more fair, democratic, decentralised than existing systems' using blockchain technology. Our review has demonstrated that blockchain technology, based on cryptographic proof instead of trust. Which allows any group of willing parties to transact directly with each other without the need for a trusted third party[12], certainly has the potential to create a more decentralised, transparent truly peer-to-peer system. This is proved by the handful of start-ups employing this technology.

Chapter 3

Protocol Design

3.1 Motivation for the proposed model

Following our review of current personal finance systems, problems associated with the model, and literature surrounding the application of blockchain. We have decided to apply token engineering methods to ROSCAs to make the process fairer, democratic, and transparent, for two core reasons.

We found those who have attempted to address the lack of access to credit in the developing world, the main operator being Kiva. However, its usage is limited and in some cases controversial [17], due to high operating fees and lack of transparency. Apart from this, ROSCAs operate on a small scale in communities. We, therefore, believe ROSCAs stand to improve massively by the application of distributed ledger technology.

Secondly, in comparison to the development of other models of personal finance, ROSCAs have been somewhat left behind. We found WeTrust is currently developing a blockchain-based platform for ROSCAs,[18] and therefore believe this model stands to gain from the application of token engineering to have its assumptions, mechanisms and system behaviours validated by simulation.

In designing crypto-economic systems, we apply theory and mechanisms from economics alongside many of the tools and concepts from computer science that enable cryptocurrencies, such as distributed systems and cryptography[21]. Using these foundations, our aim for this chapter is to design our crypto-economic protocol compromising of its necessary incentives, mechanisms,

governance model, combined with blockchain architecture in a manner consistent with the best practices of systems engineering.

In this chapter, we discuss the basic features of blockchain and the steps involved in creating a blockchain-based model of ROSCA using the token engineering process. 3.1 shows the high-level steps involved in our proposed model for decentralized ROSCA.

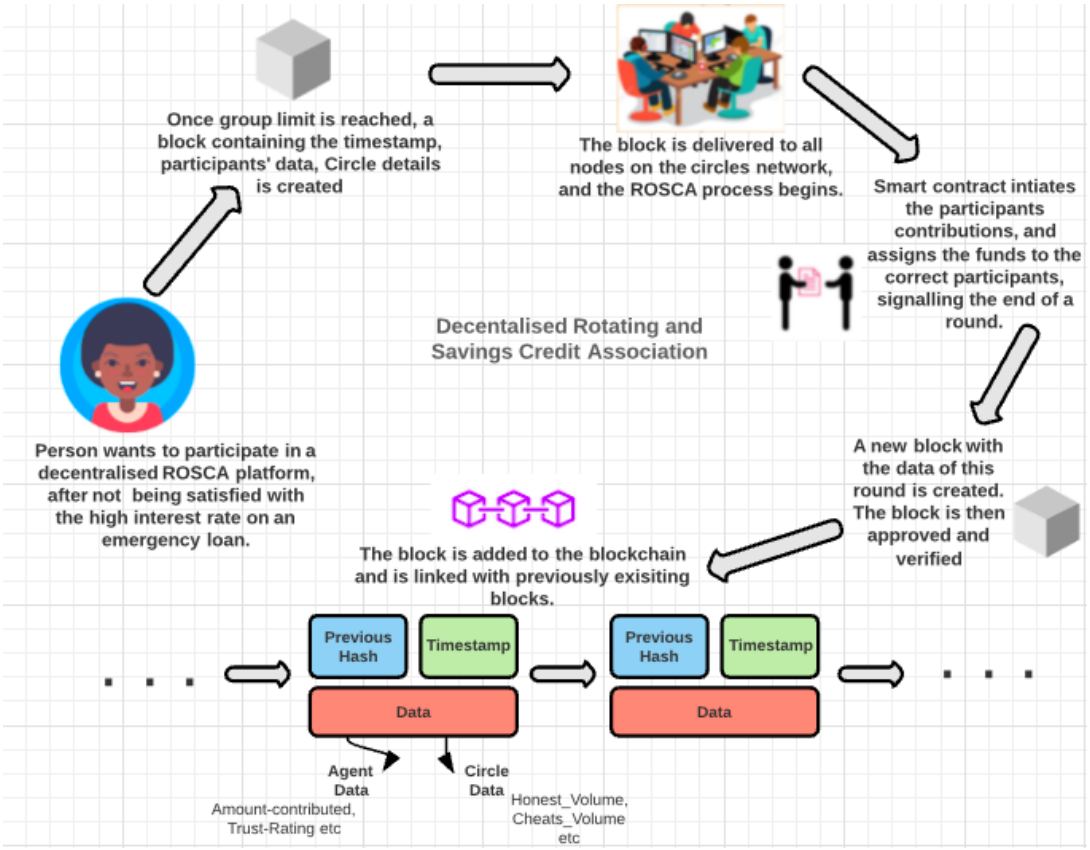

Figure 3.1: High-level steps involved in the proposed ROSCA model

3.2 System Mapping

At the start of our token engineering design journey, we begin by first understanding the goals of our system, then working backwards to determine the properties and structure required to achieve those goals[21].

At this stage of the design process we are trying to do two things:

1. Layout the system dynamics
2. Identify stakeholder groups, their possible actions, and form their incentives or individual utilities they might take.

This is will help us to identify what concepts, constructs and stakeholders are relevant to our model and to begin to define their relationships to one another, as well as our goals for the system. [22]

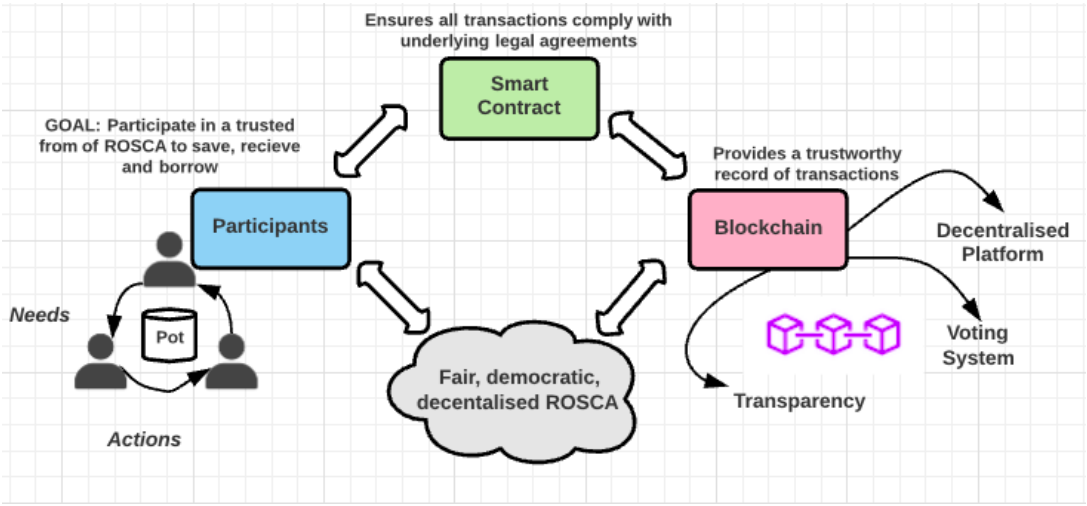

Figure 3.2: Illustration of System Dynamics

3.2.1 System Dynamics

In 3.2, we outline the dynamics and inter-connectedness of our system. We begin by identifying the purpose of our system: A 'Fair, democratic, decentralised ROSCA', we then identify three of our main structures:

1. The only stakeholder, the participants of a ROSCA circle, who collectively have their own goal of: "A trusted form of ROSCA to save, receive and borrow money". Each participant has "needs" (outlined in 3.2.4) from the platform and can take actions to achieve their goals (3.2.4).

2. Smart contract - Allows us to create rules governing a given ROSCA, then ensures all participants comply with underlying legal agreements.

3. Blockchain architecture: serves as our decentralised network, voting system (allowing participants to reach consensus), and allows a trustworthy record of transactions to be made publicly available to each node on the network.

3.2.2 System Requirements

Now that we have mapped out and understood the system's dynamics, and goals of the stakeholders as well as the system itself, we can now define concrete requirements which will facilitate

our goals.

Key functionality

1. Present the dynamics of a single ROSCA circle on a peer-to-peer decentralised network where agents share a common view of the chain.
2. Define simple strategies for agents to join a ROSCA circle (with either known or unknown peers) and receive and put money into the pot
3. Agents can choose their contribution amount and the groups duration.
4. Agents have a 'trust' rating which is a metric based on a participants' actions, which is built over time.
5. Agents are capable of utilising different strategies to maximise their reward whilst minimising their cost.
6. Agents can define rules governing their ROSCA and reach consensus with other agents to approve or disapprove borrowing.

3.2.3 Assumptions

In this model the following assumptions are made:

1. All agents have the same level of access to the real-time data on the blockchain.
2. Cheating agents (An agent who steals the pot without further contributing) are always caught and penalised
3. Agents participating in a given ROSCA circle can know one another, or not.

3.2.4 Rules and Incentive Design

The agents in these systems self-organise in response to incentives, which are chosen to motivate the behaviours needed for the protocol to function. Our challenge is to choose the right incentives, rules, and interfaces such that agents in our system, whom we do not control, self-organise in a way that fulfils the purpose of our protocol.[22]

To correctly design our system incentives, we first outline agents needs and possible actions.

Agent Needs

1. Trust-worthy record of transactions publicly available
2. Mechanisms in place in the event of defaults and fraud
3. Mechanisms in place for borrowing and allocation of funds
4. Involvement in governance and decision making

Agent Possible Actions

1. Join a ROSCA
2. Set a monthly contribution amount and be assigned a turn in which to take the pot
3. Borrow/Steal money from the pot

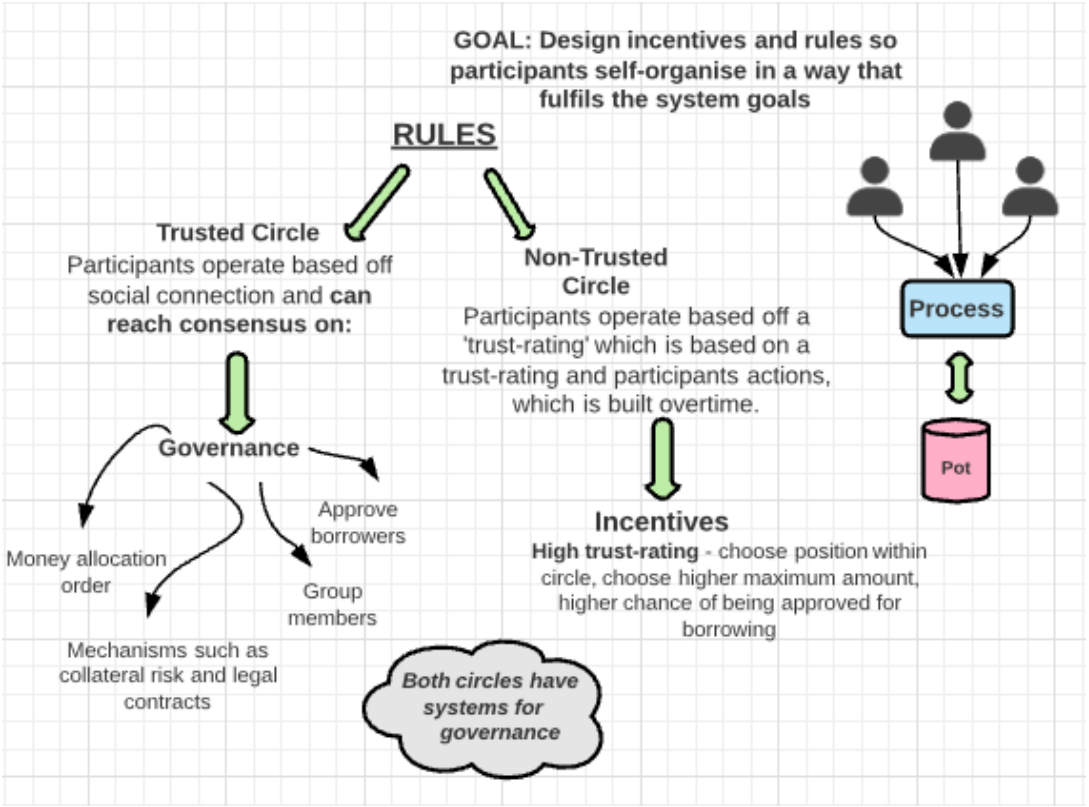

Figure 3.3: Illustration of System Rules and Incentives

The diagram above outlines the rules and incentives chosen that will motivate agent behaviours in our protocol. Simply put, there are two types of ROSCAs: one based on existing social connections without any incentives motivating behaviour (we rely on these connections to enforce behaviour), and a second (which we model) in which we attempt to influence behaviour through

our incentive system, trust-rating (3.2.4).

The system dictates the logical rules: an agent only receives as much they put in and mechanisms in place for fraudulent behaviour such as collateral risk and legal contracts. The participants reach consensus on who can borrow along with how much, whether an agent can change their standing to receive the pool earlier or later, duration of ROSCA, and maximum monthly amount.

Incentive System

Our chosen incentive is an Agent Rating System, in which we create a form of utility from being ranked by the system. As an agent contributes each month and continues contributing after receiving the pool, they accrue a personal rating of trustworthiness. The higher an agents rating, they are given priority in terms of the turn in which they receive the pot, can choose a higher monthly amount to enter and be more likely to have borrowing requests approved by the network.

3.3 Formalising the design

Now that we have a sense of the overall design of the system we can start formalising our insights using causal loop diagrams and stock and flow diagrams[21].

3.3.1 Entity relationship diagram

An entity-relationship diagram conceptualises the system and its interactions on a higher level.

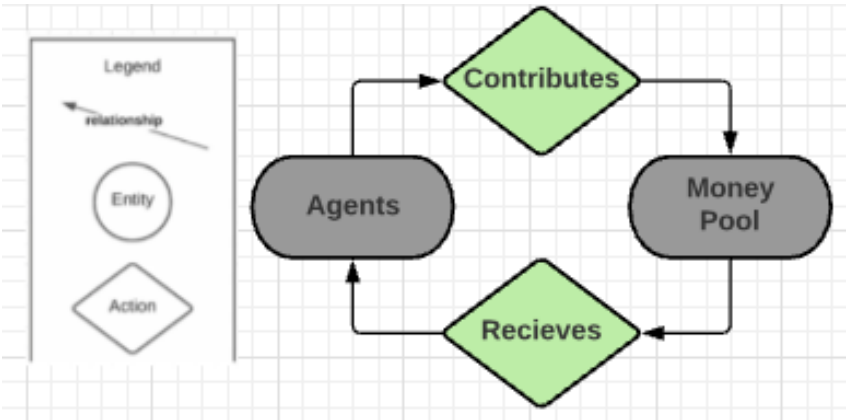

Figure 3.4: Entity Relationship Diagram

At its simplest, the system allows for two core actions: a participant contributing to the pool, and a participant receiving from the pool. This takes into account participants standing within

the circle.

3.3.2 Stock and Flow Diagram

Stocks and flows are the foundation of system dynamics modelling. A stock and flow diagram represents and encodes the mathematical structure of a model, allowing us to model some type of resource, such as money or water, moving through the different parts of a system by identifying value creation (stocks) and value sharing (flows).

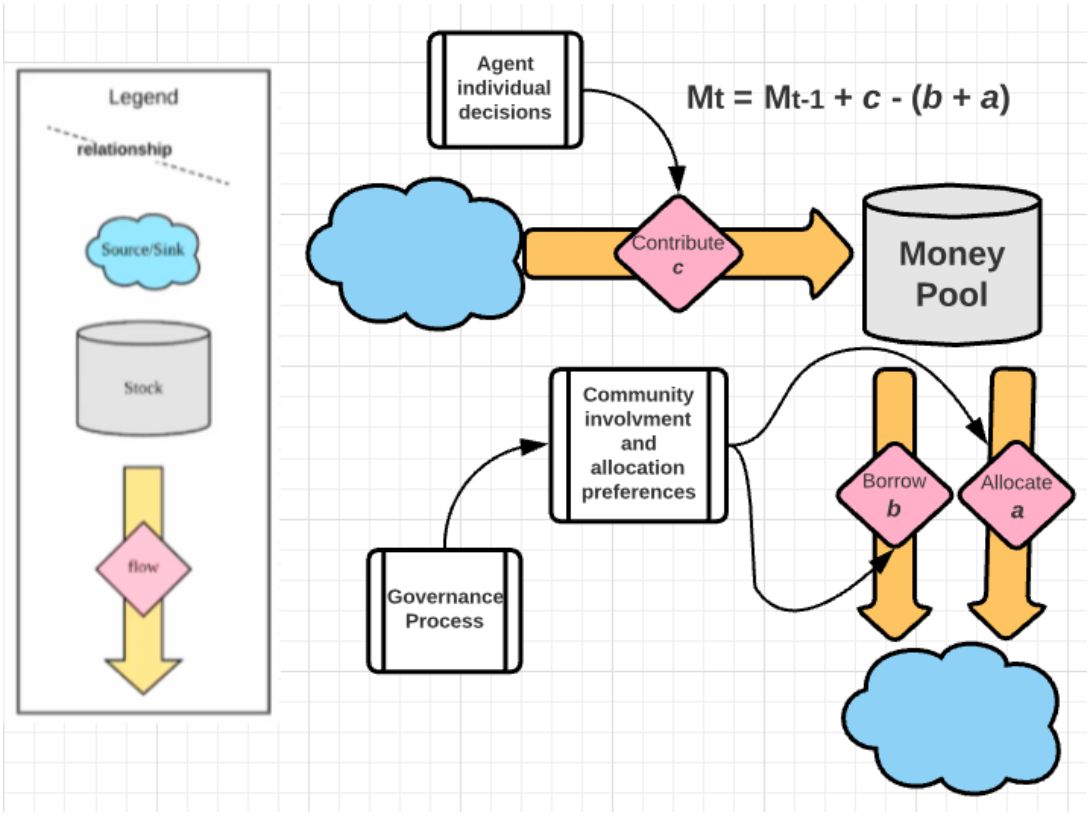

Figure 3.5: Stock and Flow Diagram

We identify a single stock, *Money Pool*, which gets its input from a flow from source C, contributions from members. We identify two more flows, $\mathbf{B}$, the total amount from borrowing agents and A, the amount allocated to a given agent. These flows are sinks of the system.

In 3.5 we also identify processes which influence the state of the flows: agent decisions and governance processes.

3.3.3 Mathematical Specification

$$\mathbf{M}_t = \mathbf{M}_(t-1) + \boldsymbol{C} - (\boldsymbol{A} + \boldsymbol{B})$$

Where:

$\boldsymbol{C} = sum(agent\_contribution)$

$\boldsymbol{A} = rosca\_duration * agent\_contribution$

$\boldsymbol{B} = total\_amount\_borrowed$

The above differential equation gives a more formal mathematical representation of the 'money pool' physical quantity. The money in the pool at a given timestep, t is equal to the amount in the previous timestep, in addition to C, the total sum of monthly agents' contribution amount, subtracted by A, then the amount allocated to a given agent:
their monthly contribution * duration and B, the total amount borrowed from the pool.

3.4 Modularising the logic and building a model

Now that we have a more formal representation of our model, and we have an understanding of states, mechanisms and behaviours of your system and the agents interacting with it, we can now map our system flows into a differential specification diagram, which will inform our cadCAD wiring in the next step of the process.[21]

Complex adaptive dynamics Computer-Aided Design (cadCAD), 4.1 is a Python library, assisting in the process of designing, testing and validating complex/economic systems through simulation[23].

3.4.1 Differential Specification

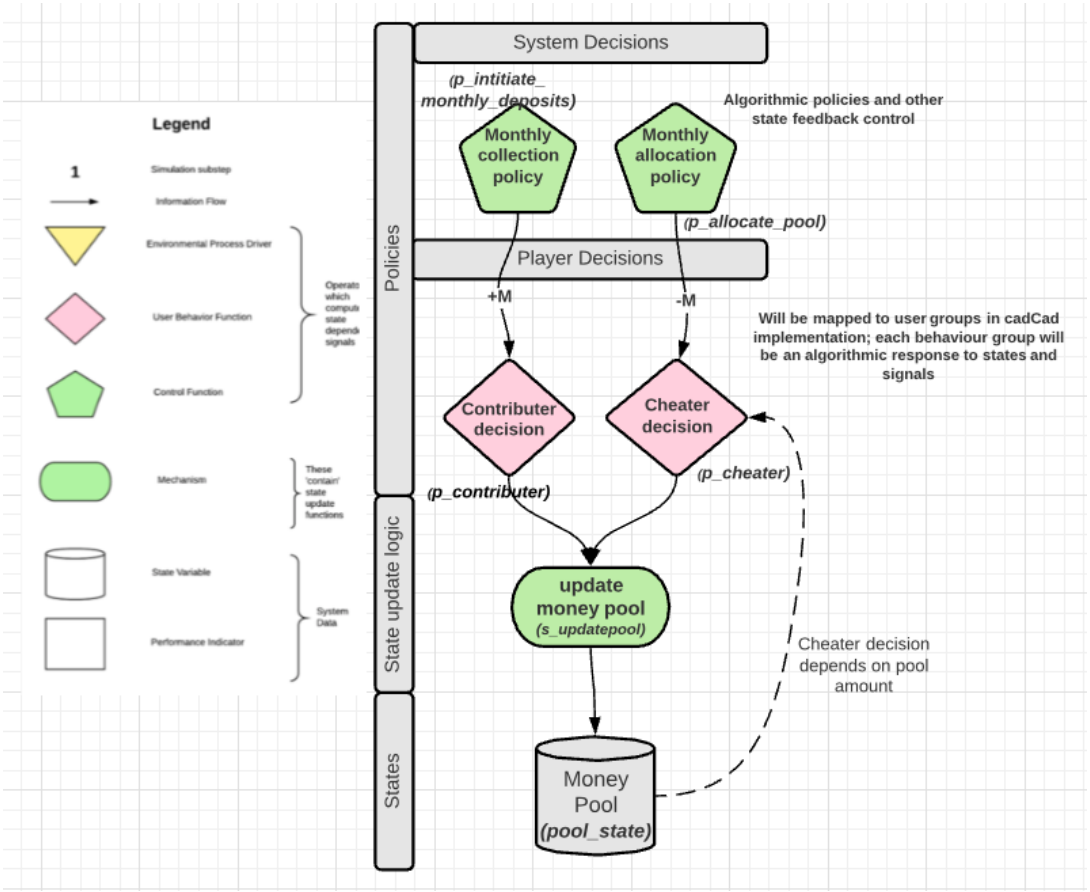

Figure 3.6: Differential Specification Diagram

The differential specification diagram takes into account system policies, agent behaviours, and exogenous processes (i.e. random external events), and the complex interaction patterns existing between them[21]. We spilt our diagram 3.6 into three rows:

1. **Policies** - determines the inputs to the system dynamics, of which we implement two types:
   (a) Agent decision: to contribute each round or to cheat and take the pot without further contributing.
   (b) Algorithmic system decision: Monthly collection and allocation policy.
2. **State Update Logic** - determines how the state of the system changes as a result of the inputs provided. We create a single mechanism that updates the *'Money Pool'* state.
3. **State** - keeps track of the interdependence of the mechanisms on the same state variables[21].

3.5 Considerations for Implementation of Blockchain

3.5.1 How can blockchain improve areas of this model?

1. **Efficiency and Automation** - Smart contract technology has predefined conditions that can dictate how the funds will flow, process claims and automate fraud detection. This substantially lowers operating costs.

2. **Decentralised Platform** – Decentralisation takes away the intermediaries and facilitates a truly peer-to-peer ROSCAs. This improves on centralised models which allows an intermediary to set rules and have unrestricted access to participants' information.

3. **Increasing Opportunities for Wealth Creation** - for those who don't have the means to reach out for traditional credit, blockchain-based ROSCAs can be an invaluable alternative.

4. **Greater Access** - Blockchain technology allows us to overcome the need for trust which creates opportunities for the creation of digital communities in addition to local communities.

5. **Incentivising Community** - Blockchain technology can be used to develop a voting system for savers, which enables them to be involved in governance.

3.5.2 Block

All transactions that take place each month in a ROSCA circle are stored in a sequence of blocks to maintain a trustworthy record. At the end of each month of ROSCA, the amount contributed by each agent, the agent who was awarded the pool along with the amount, and the final pool amount are among the information stored on the blockchain.

Each block contains an index: its position in the chain, a proof number: produced during the creation of a new block protecting the blockchain from attack, the cryptographic hash value (using the SHA-256 Algorithm) of the previous block, the timestamp of the transaction, and the data of transactions occurred that month.

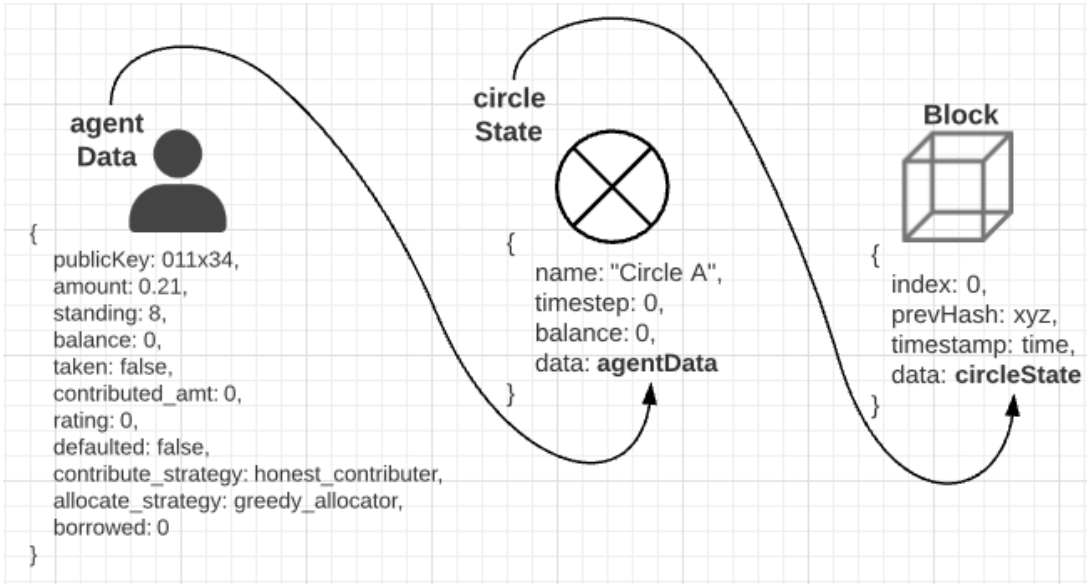

Figure 3.7: Layout of Data in a Block

Figure 5.2 outlines the layout of data that is stored in a block. The first block is created when a new ROSCA circle is initialised, *Circle State*, along with the arrival of all agents participating in the circle. The length of the blockchain will be equal to the duration of the ROSCA. The use of cryptographic hashing and hash chaining makes it difficult for anyone to tamper with information that has been already committed on the chain[24]. Every agent on the network has real-time access to the chain.

3.5.3 Digital Signature

To ensure authorship and validity of agents and their corresponding transactions, we attach every transaction (bid to borrow or receiving the pool allocated by the network) to an agent with the use of digital signatures. Each agent has a unique key pair generated. The private key used to digitally sign transactions and the public key used by others with the intention of verification.

Every transaction made to the network in the form of contributions, and money being taken in the form of allocation, or loans from the pool throughout the ROSCA duration need to be digitally signed using an agent's private key. This prevents agents from claiming they have not taken money allocated to them, or make unreasonable bids to borrow from the pool.

3.5.4 Consensus Algorithm

Consensus is a way of reaching an agreement between nodes on the network.[29] In the centralised model, platform owners decide on questions such as which transactions are legal etc. In a peer-to-peer system, there is no central governing authority to make these decisions, instead, there are different consensus protocols that serve this purpose. For example, the need verify the authenticity of the information being fed into the network by the nodes[27].

We use the Proof of Work (PoW) algorithm to reach consensus on the validity of the agents' transactions in the block. Simply, its objective after the process of verifying the transactions is to solve an arbitrary mathematical puzzle requiring a significant amount of computing work. A high level of difficulty discourages tampering with the blockchain.

For our model, we'll use a simple PoW algorithm that discourages agents from verifying fraudulent transactions.

Listing 3.1: Proof of Work Algorithm

```
def proof_of_work(last_proof):
    '''this simple algorithm identifies a number f' such that hash(ff') contain
    4 leading zeroes
         f is the previous f'
         f' is the new proof
    '''
    proof_no = 0
    while BlockChain.verifying_proof(proof_no, last_proof) is False:
        proof_no += 1

    return proof_no

def verifying_proof(last_proof, proof):

    guess = f'{last_proof}{proof}'.encode()
    guess_hash = hashlib.sha256(guess).hexdigest()
    return guess_hash[:4] == "0000"
```

Chapter 4

Simulation

Having designed, understood the components and inter-connectedness of our system, we can now simulate our model of ROSCA to be able to then validate our assumptions, and questions surrounding our model.

We split the simulation of our distributed system into three main components:

1. The abstraction of a P2P network over which agents interact 4.2
2. The addition of a rich Agent Based Model 4.3
3. Additional mechanisms to support governance 4.4

At each stage we set out our requirements to build each component, simulate then validate each component.

4.1 cadCAD

Complex adaptive dynamics Computer-Aided Design (cadCAD) is Python library, assisting in the process of designing, testing and validating complex/economic systems through simulation. It allows us to ask powerful 'what-if' questions about our system, using simulation features such as A/B tests, parameter sweeps and Monte-Carlo analysis[23].

These 'what-if' questions can concern system internal factors (eg. agent behaviours, system parameters) or system external factors (eg. market conditions, competitive activity), if a system failure has far-reaching implications cadCAD is very valuable.[23]

4.2 Abstraction of peer-to-peer network

4.2.1 Requirements

In building a blockchain-based peer-to-peer network, we set out our requirements below.

1. Present the dynamics of a single ROSCA circle on a peer-to-peer decentralised network where agents share a common view of the chain
2. Define simple strategies for agents to: **join** a ROSCA circle, be **set a contribution amount** by initiator, be **allocated the pot in a random order**
3. Understand the basic space of decision making of players in the protocol

4.2.2 Design

In this section, we go through the steps of the cadCAD Standard Notebook Layout which corresponds to our differential specification 3.6. Once this is completed, we can simulate our model and verify it satisfies our requirements (4.2.1).

1. **State Variables**

   Genesis state and genesis block

   Dynamical systems are usually rather compact to define. We need at the very least two things:

   (a) Some initial state.

   (b) The dynamics, or how state evolves over time.

   In this part, we'll focus on the initial state. We'll create a genesis state, with 10 participants whom each are set to pay 0.21ETH (£300) into the money pool for a set term of 10 months, using a randomised turn system.

Listing 4.1: Python code creating an initial state

```
# This function create an array of `AgentData` objects
def get_initial_deposits(n):
    agent = [AgentData(secrets.token_bytes(48), pounds_to_eth(300), i, 0,
    False, 0, False) for i in range(n)]
    return agent

# We then intialise a rosca circle
circle_ = initialise_circle_state("Hudas saving circle :)",
    get_initial_deposits(10))

# create initial block for the groups blockchain
genesis_state = create_block(circle_)
```

2. **System Parameters**

   System parameterisation is the process of choosing variables that impact the behaviour of the model. These parameters allow us to perform simulation techniques like parameter sweeps, Monte Carlo simulations, A/B tests and see how the system behaves under a different model parameter set.

Listing 4.2: System Parameter definition

```
system_params = {
    'duration': [10], # duration of ROSCA circle
}
system_params
```

3. **Policy Functions**

   A Policy Function computes one or more signals to be passed to State Update Functions. They describe the logic and behaviour of a system component or mechanism[21].

   In our differential specification, we identified two policies:

   (a) **A system decision** - a monthly collection from agents

   (b) **A player decision** - the decision to contribute the amount set out. For now we create a randomised agent who has a 10 percent chance of not contributing. We explore this further in our Agent Based Model (4.3).

Listing 4.3: System Policy: initiate monthly contributions (consise)

```
# system policy - initiate monthly contributions
def p_intitiate_monthly_deposits(params, substep, state_history, prev_state
    ):
    # for each agent add their monthly contribution to the collected pool
    for x in range(0, len(block_agent_data)):
        agent_contribution = agent_data[x].amount

        # player decision
        if(randomised_agent_contribution(agent_data[x], agent_contribution)
    ):
            month_sum += agent_contribution # c - from differential eq
            block_agent_data[x].contributed_amt += agent_contribution #
    update agent contributed amount
    return {'collected_amount': month_sum}
```

Listing 4.4: System Policy: Allocate to agent according to standing (consise)

```
def p_allocate_pool(genesis_state, timestep, duration, p_collected_amt):

    for x in range(0, len(agent_data)):
        if(agent_data[x].standing == timestep):
            allocation_amt = duration*agent_data[x].amount # a - from
    differential eq

            ## calculate how much is being allocated to this agent
        if(verify_allocated_amount(agent_data[x], allocation_amt,
    pool_amount)):
                agent_data[x].taken = True
                pool_amount -= allocation_amt
            else : ## not enough money in the pool for agent
                allocation_amt = pool_amount
                agent_data[x].taken = False

            agent_data[x].balance += allocation_amt
    return {'pool_amount': pool_amount}
```

4. **State Update Function**

   State evolution with block proposals

   We can now move on to defining the dynamics. Simple dynamical systems follow some rules to update their state, e.g. the orbits of planets are given by laws derived from the attraction. More complex systems are controlled: decisions made by agents in the system influence the state evolution.

   In our setting, these agents are ROSCA participants. State evolution is governed by the rules set out by the system, while participants get to decide on putting in a transaction or defaulting. The resulting state is a combination of agent decisions and state updates. How participants make these decisions are embodied by policy functions: given a state,

make a decision. At the start of the process, the Genesis State defined above is M(t-1), to update the state of the *Money Pool* to M(t) we need to calculate c - the amount collected from agents and - the amount which is allocated.

Listing 4.5: State Update - creates and adds new block to our chain

```
def s_updatepool(params, substep, state_history, previous_state,
    policy_input):
    """
    Update the pool state according to the differential equation (1):
    current_pool_state + collected_amount - allocation_amount
    """

    updateCircle_ = update_circle_state(genesis_state.data[0]['name'],
    previous_state['timestep'], p_collected_amt, p_allocated_amt,
    genesis_state.data[0]['data'])
    new_block = create_block(updateCircle_) # create block

    return 'pool_state', new_block
```

4.2.3 Validation

At this point, simulation results are returned as a list of Python dictionaries, which we then convert to a Pandas data frame. At this stage of the process, we'll manipulate and analyze our results to answer questions about our model.

Having created ten agents contributing £300 per/month on the network, with a system parameter of a duration of 10 months, we present our results below:

Table 4.1: Peer to Peer Network Simulation Results

timestep	collected amount	defaulted amount	pool balance	agent default count
1	2400.0	600.0	0.0	2
2	2700.0	300.0	0.0	1
3	2700.0	300.0	0.0	1
4	3000.0	0.0	0.0	0
5	2400.0	600.0	0.0	2
6	2700.0	300.0	0.0	1
7	2700.0	300.0	0.0	1
8	2400.0	600.0	0.0	2
9	2400.0	600.0	0.0	2
10	2700.0	300.0	0.0	1

From the table above, we see two things;

1. Randomness in the number of times and when an agent defaults, illustrated in 4.1

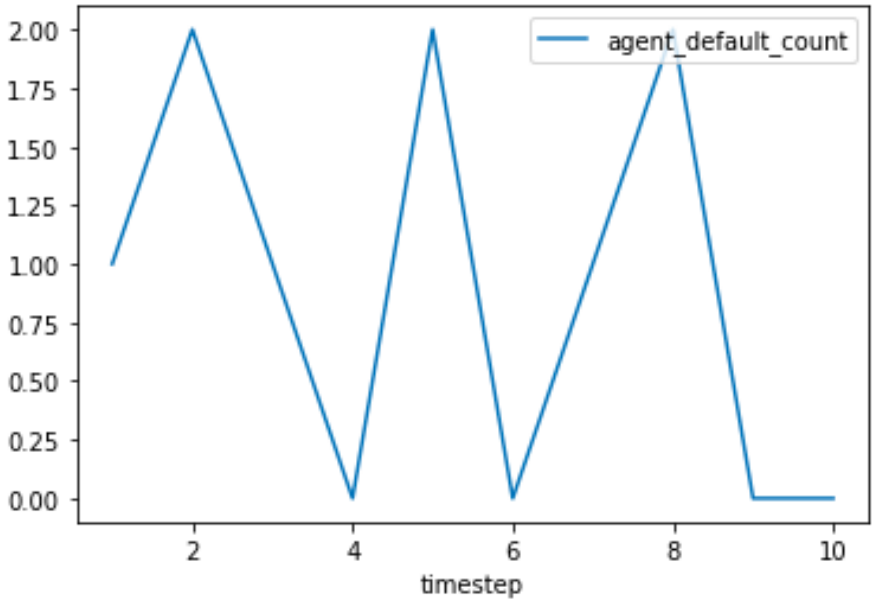

Figure 4.1: Agent Default Count

2. Given we've set the amount given to the agents monthly as £3000, we see the number of times an agent receives their full payout is low compared to the number of times they don't, illustrated in 4.2

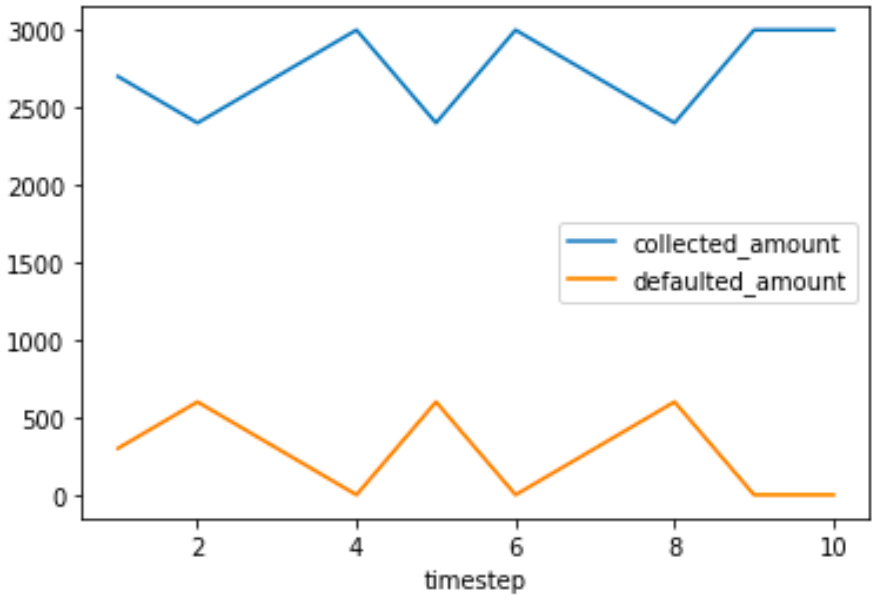

Figure 4.2: Collected Amount Compared to Defaulted Amount

Model Appraisal

Looking at the requirements we set out (4.2.1), we have successfully created a simple model which presents the dynamics of a ROSCA circle on a blockchain network where participants can carry out two core actions: contributing to the pool and receiving from the pool.

How can we extend this model to introduce additional complexity to more fit real-world implementations?

- We can define specific agent behaviour and test what happens under different scenarios.
- Agents in this system are not incentivised or dis-incentivised.
- Design mechanisms outlined requirements voting and borrowing, the building of our core mechanisms defined here.

Our next step is to design incentives and agent strategies to ensure agents behave in a way that fulfils protocol.

4.3 Agent-based Model

Since we are modelling a social and economic system, we are constantly asking what our agents will do. Answering this is a second step towards an abstracted (but realistic) simulation environment, where we essentially outline the economic incentives associated with the choices the agents can make. More specifically, what economic incentives are associated with the choice to behave honestly, or dishonestly.

In the worst-case scenario, multiple agents from the same circle behave dishonestly, resulting in the rest of the agents not receiving the full amount they contributed (4.2) or disincentivising them from continuing to contribute. How can we create policies that are robust to network uncertainty and strategic interactions?

We will develop these questions throughout, getting insight from our models and simulations.

4.3.1 Requirements

For this version, we extend our ROSCA model to focus on two things:

1. Producing rich agent-based models
2. Programming incentives and seeing how agent behaviours map to these the incentives (rewards, penalties, general outcomes)

In an attempt to understand agent actions, and define realistic strategies and behaviours in our differential game, we utilise Game theory. Game theory is the study of strategic situations where players choose different actions in an attempt to maximize their payoffs which in turn are determined by the choices of other individuals.[25]

Agents' Dilemma

In this simulation the agents are faced with two choices:

1. To cheat and steal the pot, or behave honestly and continue contributing after the pot has been allocated to them. An agent is faced with this choice once for the entire duration - we call these agents allocators.
2. To contribute money, or not. An agent is faced with this choice every timestep (month) - we call these agents contributors.

We explore these choices using a differential game. In game theory, differential games are a

group of problems related to the modelling and analysis of conflict in the context of a dynamic system.[26]

Assuming the choice to steal the pot is expensive (incurs a heavy cost) and agents do not undertake this task unless their is an expected return-on-investment (ROI) is greater than penalties incurred or the incentives associated with good behaviour such as a higher trust rating.

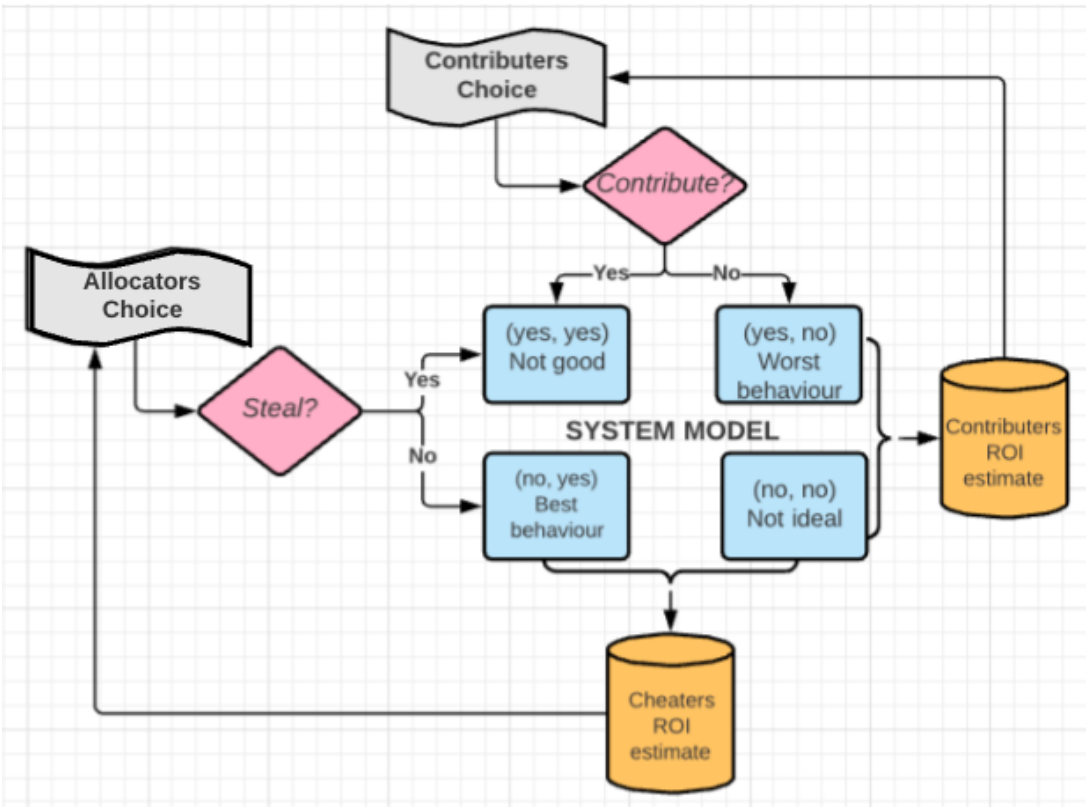

Figure 4.3: Agents' Payoff Matrix

Evaluating Agent Strategies

We begin by constructing a payoff matrix (4.3). We see the strategy that results in the optimal outcome for the system dynamics, is for each agent to **contribute** their share each month and not **steal the pot** when they are allocated.

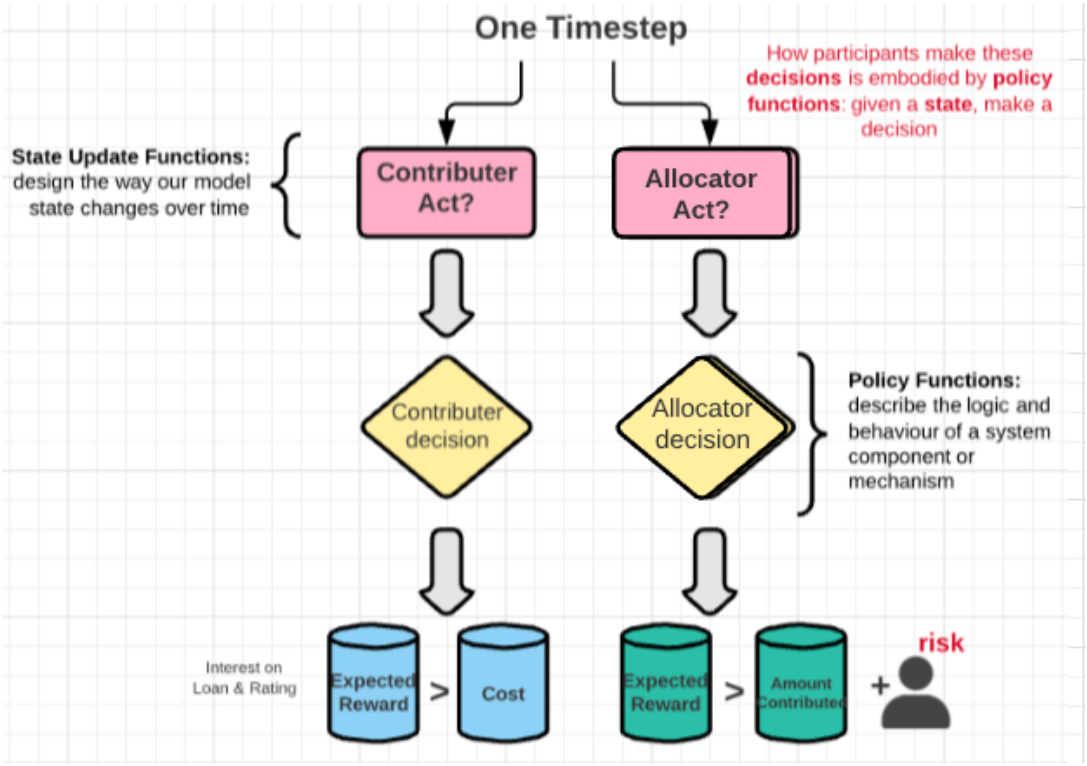

Figure 4.4: Agent Decision

At each point in time, both Contributor's and Allocator's must make a decision about how to act, which is made by weighing their estimated reward compared to their cost, outlined in 4.4. Contributor's and Allocator's beliefs about the expected returns on their actions are estimated by their observations of the others' actions over time. For example: if an agent sees others either aren't contributing or have stolen the pool, this will heavily influence their decision on whether to continue behaving honestly.

4.3.2 Design

In this section we go through the steps of the cadCAD Standard Notebook Layout, corresponding to our differential specification (3.6), which has been modified to reflect the simulation of an Agent-Based Model.

1. **Agent Strategies Definition**

   Here we define our behavioral models in which agents interact according to their own heuristic strategies, illustrated in 4.5 and defined in Listing 4.6 and Listing 4.7

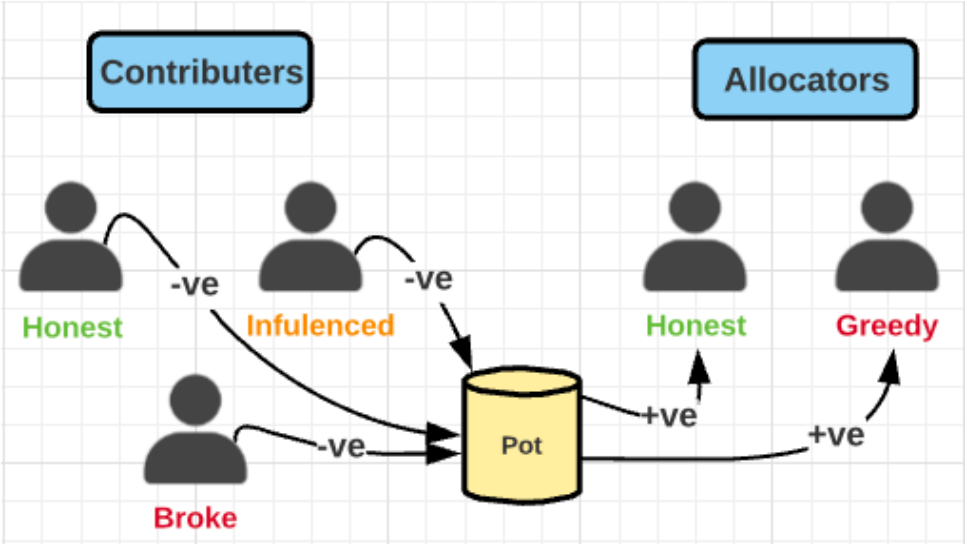

Figure 4.5: Agents' Strategies Illustration

We define three strategies for contributors:

Listing 4.6: Python code defining contributor strategies

```
# contributes just as long as cost isn't greater than reward
def honest_contributer(reward, cost):
    if(cost > reward):
       return False
    else :
        return True

# gets discouraged from contributing if cost > 25% of reward
def influenced_contributer(reward, cost):
    if((0.25 * cost) > reward):
       return False
    else :
        return True

# 5% chance this contributer doesn't have enough money to contribute
def broke_contributer(reward, cost):
    if(random.random() < 0.05):
       return False
    else :
        return True
```

We define two strategies for allocators:

Listing 4.7: Python code defining allocator strategies

```
# doesn't steal pot as long as cost is not greater than reward
def honest_allocator(reward, cost):
    if(cost > reward):
       return False
    else:
        return True

# if reward is greater than cost or agent accepts risk of not being caught
def greedy_allocator(reward, cost):
    risk = random.random()
    if((reward > cost) or (risk < 0.5 and reward != 0)):
        return True
    else:
        return False
```

2. **State Variables**

Listing 4.8: Agent Based Model Genesis State

```
genesis_state = {
    'Contributers_On': True, # contributers are active
    'Cheaters_On': False, # cheaters are inactive
    'Total_Volume': 0, # total volume of pool activity (GBP)
    'Honest_Volume': 0, # volume of honest (contributed) activity (GBP)
    'Dishonest_Volume': 0, # volume of dishonest (non-contributed) activity
    (GBP)
    'Cheats_Volume': 0, # volume of cheater activity (GBP)
    'Contributers_Rating': 0, # ratings of contributing agent
    'Contributers_Cost': 0, # cost incurred by contributers
    'Contributers_Reward': 0, # rewards collected by contributers
    'Cheaters_Cost': 0, # costs incurred by cheaters
    'Cheater_Reward': 0, # rewards (profit) achieved by cheating
    'timestep': 0,
    'agent_data': agent_genesis, # function to create agents and assign
   them strategies
    'Duration': 10 # duration of circle
    }
```

We create our initial simulation state by creating agents, which is stored in our genesis state above, initialising a ROSCA circle, creating our initial block and storing our genesis state above, as seen previously in, 1.

In this simulation, we want to pay particular focus to the volume of specific types of transactions (hence the data layout, 2), we also assign strategies (4.5) to our agents.

3. **Reward and Cost**

   Here we outline our State Dependant Reward and Cost Functions which characterise the players' utilities.

Listing 4.9: Python code defining agents reward and costs

```
# contributers's reward per round contributed => amount agreed
def contributers_reward(data, s):
    return calculate_agent_amount(data, s['timestep'], s['Duration'])

# Contributer's cost per round cheater acts => amount defaulted if at all
def contributer_cost(s):
    share = 0.1
    return (share * s['Cheats_Volume'])

# Cheater's reward per cheat
def cheater_reward(s, allocation_amt):
    if(s.contributed_amt >= allocation_amt):
        return 0.0
    else :
        return allocation_amt - s.contributed_amt # Pot amount -
   Contributed amount

# Cheater's cost per cheat caught
delta = 0.2
def cheater_cost(s, allocation_amt):
    return (delta * allocation_amt) + allocation_amt - s.contributed_amt
```

   To put simply, a contributing agents reward is the amount they receive that they have not yet contributed, and if an agent is allocated the pot and no longer contributes, this is a *cost* shared between the contributors.

   Similarly, the reward for an agent that is allocated the pot and decides to cheat is the amount they have received, subtracted by anything they have contributed so far. Their cost is the amount reward in addition to a penalty of 0.2 of this amount.

4. **Agent Behaviours Definition**

   We can now define our two types of agents

   (a) **A Contributor**

   Contributer motivated to contribute their share if reward is greater than the cost of other agents stealing

Listing 4.10: Contribuing agent definition

```
def p_contributer(state, agent):
    act = False
    cost = contributer_cost(state)
    reward = contributers_reward(agent, state)

    if (agent_strategy(reward, cost)):
        act = True
        agent.rating += 0.2 # increase agent rating
        state['Contributers_Rating'] += 0.2 # increase global rating
        agent.contributed_amt += agent.amount # update agent
    contributed amount

        return act
```

(b) **An Allocator**

An allocator is motivated to steal the pot if the reward is greater than their amount contributed so far and can take the risk of being caught

Listing 4.11: Allocating agent definition

```
def p_allocator(state, agent, allocation_amt):
    act = False
    reward = cheater_reward(agent, allocation_amt)
    cost = cheater_cost(agent, allocation_amt)
    agent_strategy = agent.allocate_strategy

    if(agent_strategy(reward, cost)):
        act = True
        state['Cheats_Volume'] += reward
        state['Cheater_Reward'] = reward
        state['Cheaters_Cost'] = cost
    else :
        state['Contributers_Reward'] = reward

        return act
```

5. **Policy Functions**

In this section, Policies are defined to initiate our cheaters and contributors choices.

In our differential specification, we focus on two policies:

(a) A system decision - Monthly collection policy. This initiates a players decision - the decision to contribute the amount set out or not. (3a)

(b) A system decision - Monthly allocation policy. This initiates a players decision - the decision to steal the pool allocated to you or not. (3b)

6. **State Update Functions**

   So far we have described the system of rewards and the simple strategies being evaluated, but in order to implement a differential game we also explicitly encode the mechanisms. Mechanisms are the functions that map decisions to changes in the system state[35].

Listing 4.12: State update function

```
def s_updatepool(params, substep, state_history, previous_state,
    policy_input):
        return 'pool_state', genesis_state
```

4.3.3 Validation

Our simulation results are returned as a list of Python dictionaries, which we then convert to a Pandas data frame. At this stage of the process, we'll manipulate and analyze our results to answer questions about our model. We keep our system parameters the same, with ten agents contributing £300 per/month, and a duration of 10 months we present our results below.

Table 4.2: Agent Based Model Simulation Results

Timestep	Contributers On	Cheaters On	Total Volume	Honest Volume	Dishonest Volume
0	True	False	0.0	0.0	0.0
1	True	False	3000.0	3000.0	0.0
2	True	False	3000.0	3000.0	0.0
3	True	True	3000.0	3000.0	0.0
4	True	False	3000.0	2700.0	300.0
5	True	True	3000.0	2100.0	900.0
6	True	False	3000.0	2400.0	600.0
7	True	True	3000.0	2400.0	600.0
8	True	False	3000.0	2100.0	900.0
9	True	False	3000.0	2100.0	900.0
10	True	False	3000.0	1800.0	1200.0

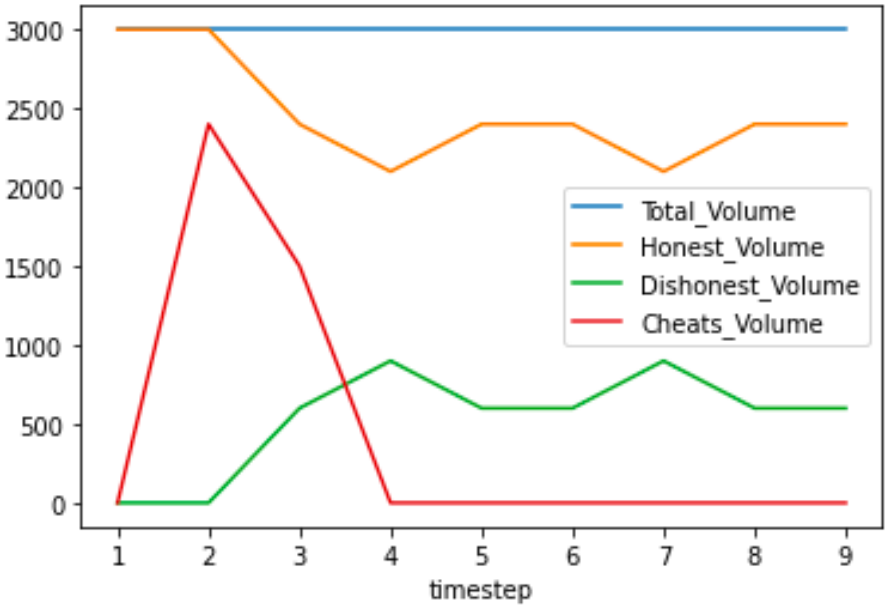

Figure 4.6: Volume of Types of Activity

Here we observe the evolution of the three different types of volumes over time. We see two distinct things:

1. The agents are overall inclined towards behaving honestly - with the honest volume somewhat stable
2. Following an act of a single agent cheating and stealing the pot (without contributing further) disincentivises the other agents and results in the following months to have an increase in Dishonest behaviour (agent not contributing)

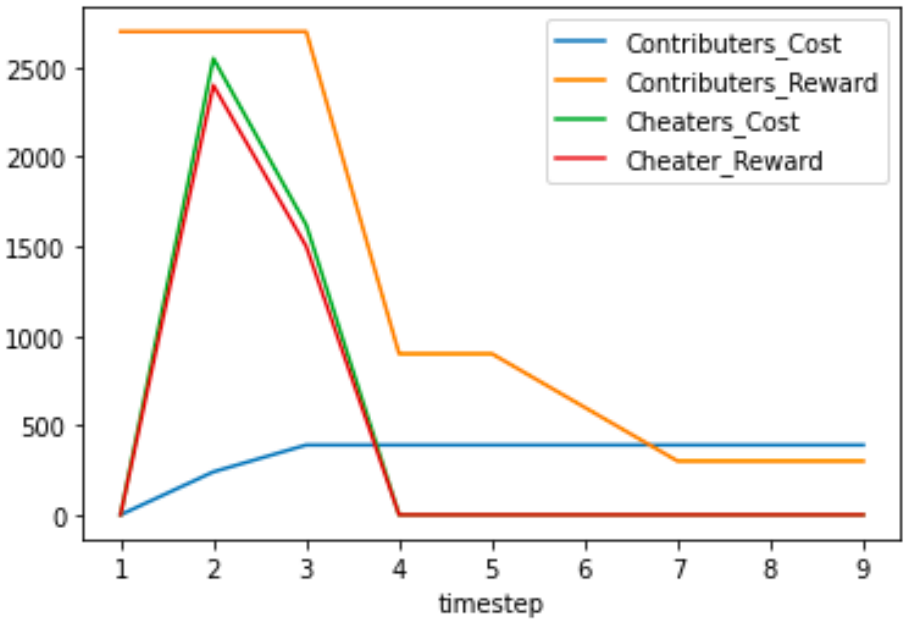

Figure 4.7: Reward and Cost for Different Agents

Here we map out our contributors and cheaters costs vs rewards. We see:

1. The contributors that benefit from this scheme the most are those that receive the pot at the beginning of the ROSCA circle, towards the end an agent would've contributed almost all/all the money they put in.

2. Similarly, a cheater benefits if they decide to steal at the beginning of the circle, however, this is negligible as the cost of getting caught is always higher.

3. Cheating negatively impacts the entire circle as they all share the cost of the amount that was stolen, however, this is almost always negligible also, as cheaters face a penalty when caught

In the section below, we use the pandas built-ins to compute some extra fields to explore the financial flows.

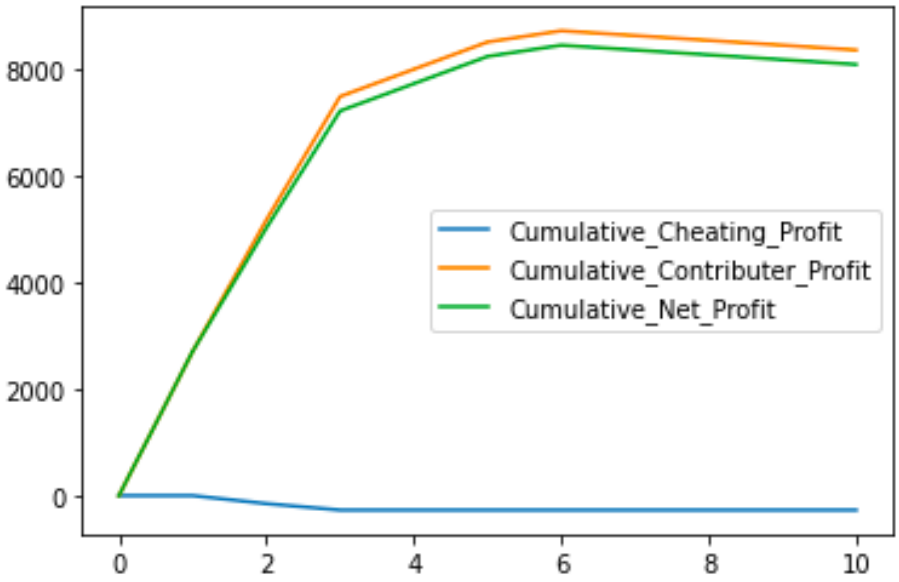

Figure 4.8: Cumulative Profit for Agents

Based on the assumption that cheaters will always be caught, and they are charged a penalty for cheating, the Cumulative Contributor Profit is significantly higher than the Cheaters.

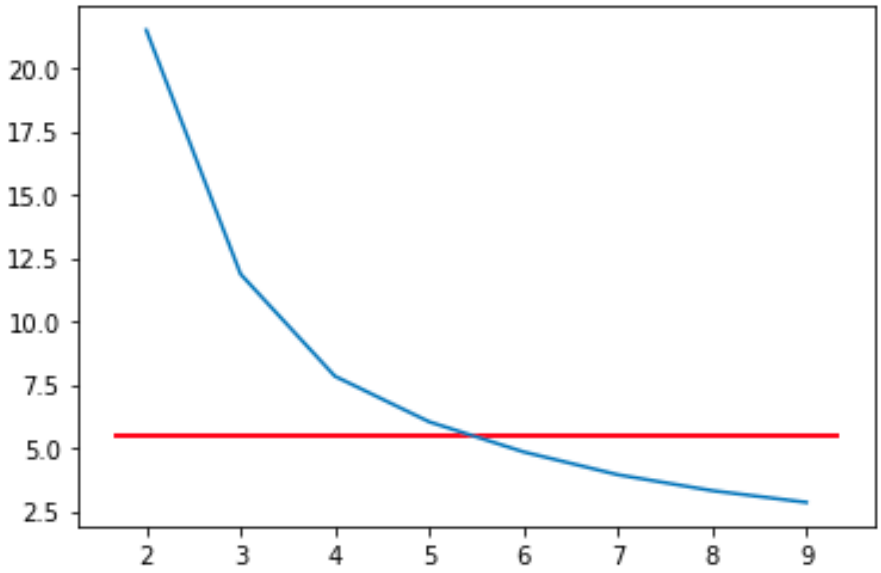

Figure 4.9: Cumulative Contributor Return on Investment

This graph supports (1) in that the agents who benefit the most from this scheme are those who receive the pot in the first few months.

Model Appraisal

This chapter introduced a key element of distributed systems: the addition of a rich agent-based model. We programmed a range of different types of agents in different scenarios: honest contributor, influenced contributor, greedy allocator etc, and economic incentives associated with the choices. All agents on the network behave in a way that maximises their reward while minimising their cost.

In this simulation, we observed different types of agents interact with each other, eg - if one agent steals the pot how does this influence the actions of others in the circle. We introduced incentives and rules which serves to influence decision making in the protocol, eg - an agents decision not to contribute only affects their ROI and rating, therefore in the interest of maximising your pot it is better to contribute.

How can we extend this model to introduce additional complexity to more fit real-world implementations?

- Enforce our incentivisation system, by allowing agents to be prioritised in the system using the trust-rating system implemented.
- Allow agents to choose their contribution amounts.

While we have so far built a transparent, decentralised peer-to-peer ROSCA model, we haven't met our goal of our system being democratic. While agents are free to make their own decisions, there is no system for governance. This is our next and final step.

4.4 Governance

4.4.1 Requirements

Our goal for this final version is to complete our distributed system by building a system of governance, we want to:

1. **Build a mechanism for voting**

   A system of voting allows the continuous organising of a communities preferences into discrete decisions to allow the management of that communities resources. Suppose a group of people want to coordinate to make a collective decision. Social dynamics such as discussions, signalling, and even changing one's mind based on feedback from other's input play an important role in these processes. A voting mechanism augments these social dynamics, making group preferences more transparent.

2. **A mechanism for borrowing from the pot**

3. **To further complete our system to fit real-world implementations, we allow agents to:**

   (a) Choose their contribution amount

   (b) Receive and be able to borrow from the pot according to their 'trust rating

4.4.2 Design

In this section, we go through the steps of the cadCAD Standard Notebook Layout which corresponds to our differential specification (3.3). Once this is completed, we can simulate our model and verify it satisfies our requirements set out in 4.4.1

1. **System Parameters**

Listing 4.13: System Parameterisation

```
system_params = {
    'duration': [5, 8, 12],
    'participants': [5, 8, 12],
    'max_amount': [2, 4, 6], #  Hundreds
}
system_params
```

In this simulation, we define a range of parameters that we want to use for parameter sweeps (2) are:

- A parameter representing the number of agents participating in a given circle. We sweep three values: 5, 8, 12 which simulates a range of sizes of circles. These values need to mirror duration values (each month one participant receives pot)
- A parameter representing the maximum limit in GBP a given participant is allowed to enter for monthly. We sweep two values: 200, 400 and 600

2. **State Variables**

As before, we create our initial simulation state by creating agents, initialising a ROSCA circle, creating our initial block which stores our initial data. However, for this simulation, we allow agents to choose their contribution amount and receive the pot according to their 'trust rating', illustrated below.

Listing 4.14: Function which creates an object of agents.

```
def get_initial_deposits(max_amount, n):
    agent = [AgentData(secrets.token_bytes(48), choose_amount(max_amount),
    i, 0, False, 0, False, 0, assign_contributing_strategy(i),
    assign_allocating_strategy(i), 0)
              for i in range(n)]

    ordered_agents = order_agent_standing(agent)  ## according to their
    trust-rating
    return ordered_agents
```

3. **Voting mechanism**

Allow agents to define rules governing their ROSCA and reach consensus on whether or not to approve agent borrowing request.

Listing 4.15: Mechanism for Voting based on circle rules eg: participants history

```
def approve_borrowing(params, state, agent_no, amount):

    threshold = (params['participants'] * 0.8) ## threshold for a vote to
    be approved - 80% of group
    participants = state['Agent_Data']
    approve = False

    if(get_monthly_default_count(participants, agent_no) != 0):
        message = "Agent borrowing not approved: Has a defaulting record."
    elif(amount > participants[agent_no].amount):
        message = "Agent borrowing not approved: Amount requested to high
    for agent."
    elif(participants[agent_no].rating < 0.5):
        message = "Agent borrowing not approved: Trust rating too low."
    else:
        message = "Agent borrowing approved: Amount - " + str(amount)
        participants[agent_no].borrowed = amount
        approve = True

    rejected_count = partcipant_vote(participants)

    if(params['participants'] - rejected_count < threshold) :
        return {False, "Agent borrowing denied: Participants vote" }
    else:
        return {approve, message}
```

We create a simple mechanism for voting above which is based on system rules. There is then a participant vote conducted that could either support the decision or override and reject it based on a voting threshold.

4. **Borrowing mechanism**

Listing 4.16: Mechanism for Borrowing

```
def p_agent_borrow_request(params, substep, state_history, previous_state):

    amount = choose_amount(params['max_amount']) # agent chooses amoun to
    borrow
    borrowed_amt = 0.0

    if(state['Surplus_Volume'] != 0 and random.random() < 0.5): # 50%
    chance of certain agent asking to borrow from pool
        approved, message = approve_borrowing(params, state, agent_pos,
    amount)
        print(message)

        if(approved):
            borrowed_amt = amount
            state['Borrowed_Volume'] = amount
            state['Surplus_Volume'] -= amount # decrement global pool value

    return {'borrowed_amount': borrowed_amt}
```

Allows agents to put forward a borrowing request from pool.

5. **State Update Function**

   So far we have created a mechanism that allows an agent to request to borrow from their pool, and a mechanism allowing agents to reach consensus on whether to approve or not. Our state update function (4) defines our system evolution

Having now completed our requirements outlined throughout the chapters, we now turn to the task of validating and testing our system as a whole, using A/B testing, parameter sweeps and Monte Carlo analyses.

Chapter 5

Simulation Analysis

In outlining our project aims and objectives (1.1), we identified two main questions which we will ask and in turn use to evaluate our design and stimulation. First, how well does our system incentivise agents, whom we do not control, to self-organise in a way that fulfils the purpose of our protocol? And secondly, Is my proposed model of personal finance more fair, democratic, decentralised than existing systems?

To answer these questions, we first need to validate and test our system. We evaluate our model by:

1. Conducting computational experiments (What-if questions) using simulation
2. Validating whether or not our model has met our requirements previously set out.

We use the following cadCAD Simulation Methods in our experiments:

1. **Monte Carlo Method**

   Monte Carlo methods, or Monte Carlo experiments, are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results[36]. The underlying concept is to use randomness to solve problems that might be deterministic in principle[37].

   Non-determinism

   In computer science, a non-deterministic algorithm is an algorithm that, even for the same input, can exhibit different behaviours on different runs, as opposed to a deterministic algorithm.

In our simulation we defined three stochastic processes:

(a) The likelihood of an agent assigned with 'broke contributor' strategy to default

(b) The risk-taking level of an agent assigned with 'greedy allocator' strategy to steal the pot

(c) The chance of an individual agent putting forward a request to borrow from the pot

2. **Parameter Sweeps**

   Simulation method that tests a range of parameter configuration choices. By careful selection of what parameter values you choose to change, and how you choose to change them, with a parameter sweep you can examine what effect they have on system dynamics[23].

3. **A/B Testing**

   In experiment methodology that tests two or more model variants against each other. By not clearing the cadCAD config state at each stage, we end up with a set of simulation results[23].

5.1 What-if Matrix

Our model attempts to answer the following questions:

1. Given agents with different strategies always work to maximise their return, how often do agent behave **honestly** vs **dishonestly**
2. Does the system overall **reward** honest agents over dishonest agents whilst keeping their **costs** to a minimum?
3. How does increasing the **number of agents** and **duration** of the ROSCA impact the outcomes. Is there are increasing/decreasing returns with an increase in participation in terms of governance, or are agents more contribute/not contribute?
4. How does an increase in amount agents are allowed to enter with impact the outcomes of the ROSCA circle?

Table 5.1: What-if matrix

What-if..	Type of experiment	Variables/Parameters	Values/Ranges to be tested
1 (1)	Monte Carlo runs	D=10, P=10, A=500	
2 (2)	Monte Carlo runs	D=10, P=10, A=500	
3 (3)	A/B + Parameter Sweeps	participants and duration	5, 8, 12 participants and duration
4 (4)	A/B + Parameter Sweeps	max amount	200, 350, 600GBP per/month

5.1.1 Analysis 1

Given agents with different strategies always work to maximise their return, how often do agent behave honestly vs dishonestly?

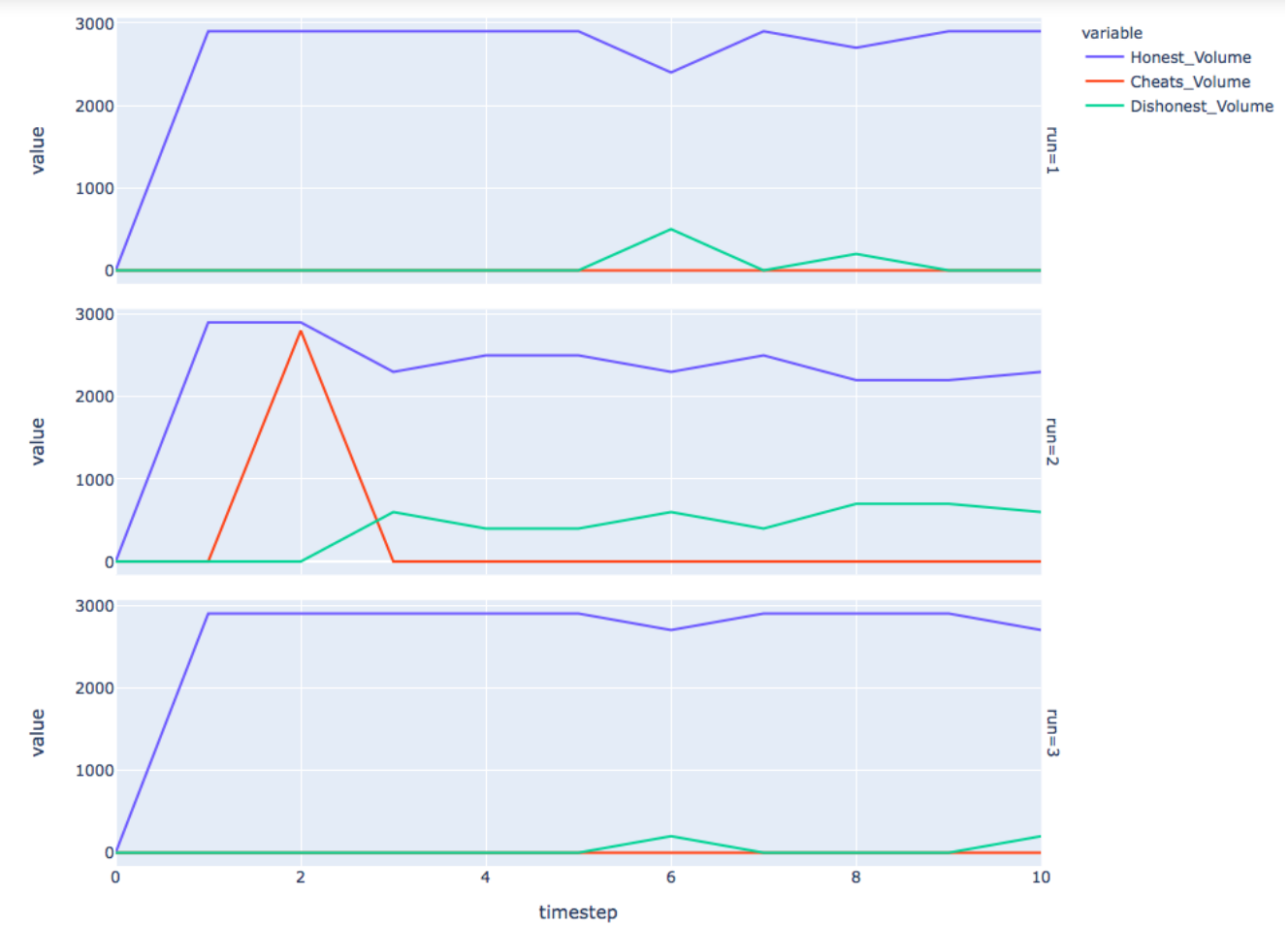

Figure 5.1: Monte-Carlo analysis for Honest and Dishonest Volumes

Experiment Results

We run three Monte-Carlo runs, with our parameters the same: Duration: 10 months, Participants: 10, Max Amount: 500GBP Per Month. The graph tells us a couple of things:

It further cements the idea of agents only stealing the pot, when its worth their while (at the beginning when they haven't contributed as much) While agents may be dishonest on random occasions, they are on average on Honest. However, a single agent stealing money from the pot causes an increase in agents defaulting and being Dishonest in their contributions.

Therefore, with this insight, we can answer the question posed in the analysis: Agents as a whole behave honestly, both when it is their turn to contribute and when they are allocated the pot. However, a single agent stealing the pot negatively influences the agents and results in agents not contributing and others then not receiving their full amount.

In terms of designing the protocol to minimise this effect, it is imperative to order the standing in which agents receive the pot with proven Honest agents first and those most likely to Cheat towards the end where they will not be motivated to steal. This is essentially a description of our incentive system.

5.1.2 Analysis 2

Given agents with different strategies always work to maximise their return, how often do agent behave honestly vs dishonestly?

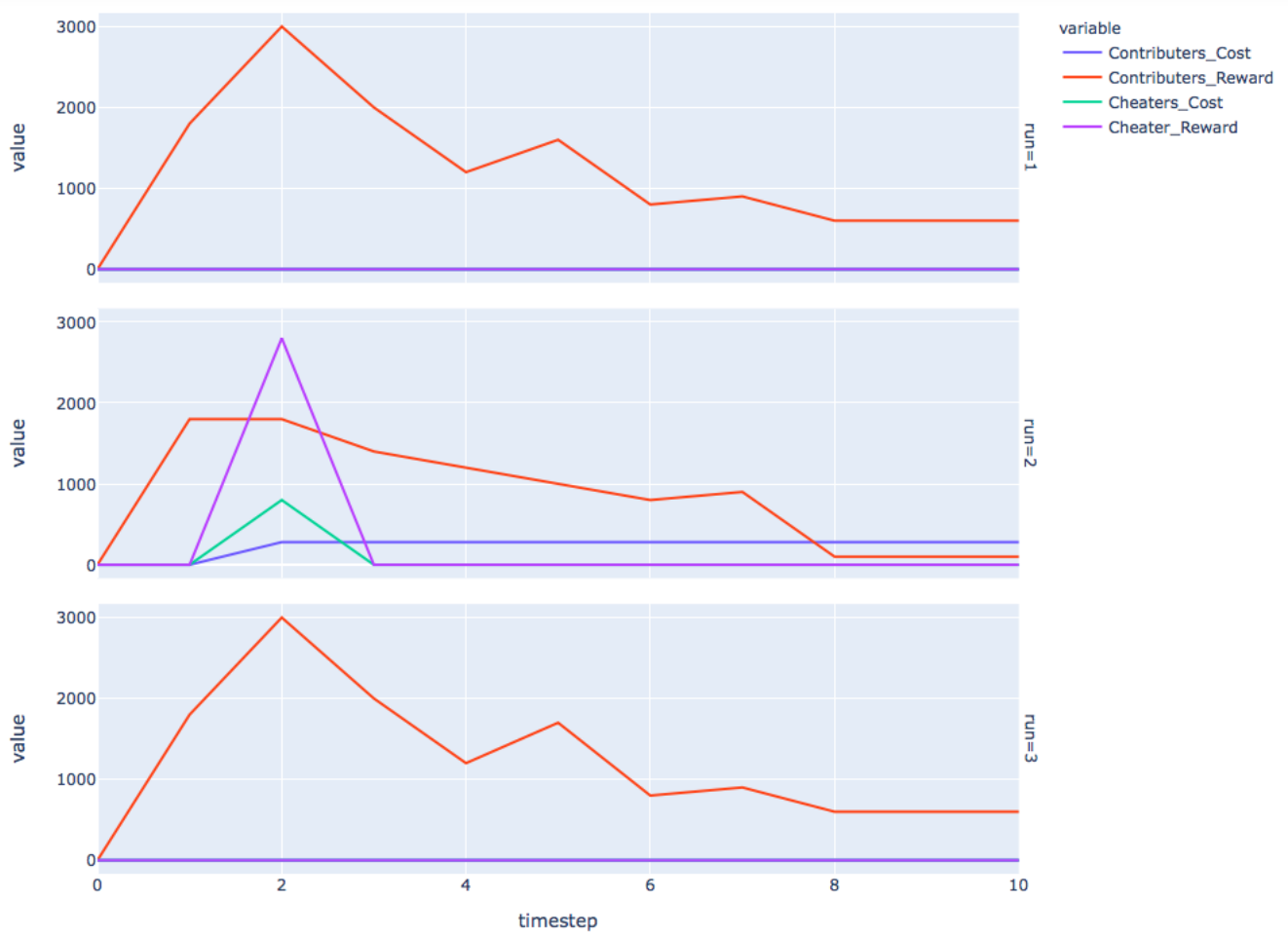

Figure 5.2: Monte-Carlo analysis illustrating agent Reward and Cost Volumes

Experiment Results

We run three Monte-Carlo runs, with our parameters the same: Duration: 10 months, Participants: 10, Max Amount: 500GBP Per Month. The graph tells us a couple of things:

We see agents who contribute benefit the most from the scheme, particularly those who have a standing which is early on in the circle, whilst only incurring a cost if an agent cheats and steals the pot.

5.1.3 Analysis 3

How does increasing the number of agents and duration of the ROSCA impact the outcomes. Is there are increasing/decreasing returns with an increase in participa-

tion in terms of governance, or are agents more contribute/not contribute?

For an example:
A longer duration of a circle may lead to participants engaging in group and don't cheat as much.

To conduct this analysis, we need to simulate three different versions of the same system, by sweeping three values for Duration and number of Participants, namely:

1. ROSCA circle with 5 agents with a duration of 6 months and a maximum monthly amount of 500
2. ROSCA circle with 8 agents with a duration of 12 months and a maximum monthly amount of 500
3. ROSCA circle with 12 agents with a duration of 20 months and a maximum monthly amount of 500.

   We will use the results to answer the question posed.

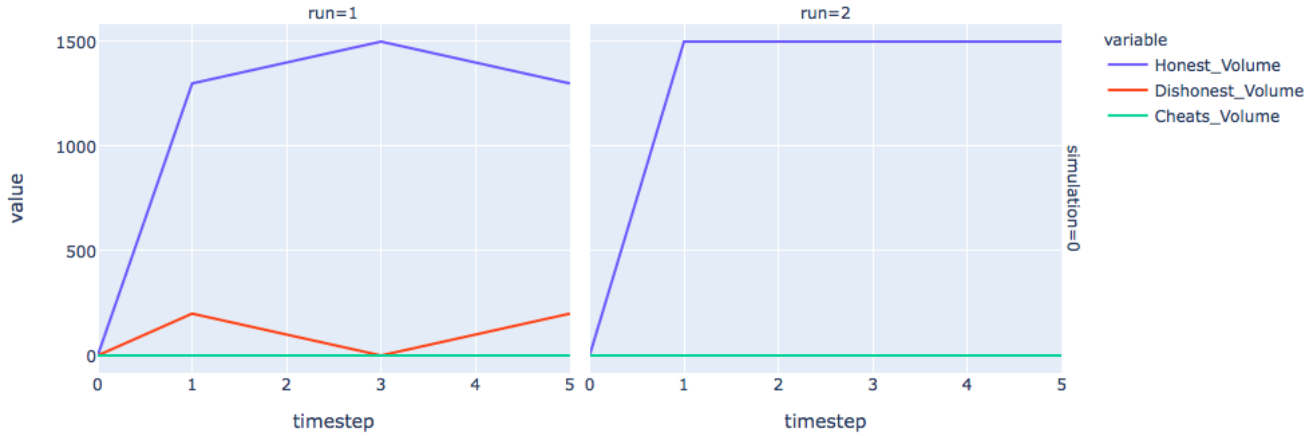

Figure 5.3: Simulation with parameters duration and participants at 5

Figure 5.4: Simulation with parameters duration and participants at 8

Figure 5.5: Simulation with parameters duration and participants at 12

Experiment Results

We run two Monte-Carlo runs for each simulation to obtain a second set of results for each parameter sweep. From the three simulations above we note a few things:

1. Simulation 1 - When few agents are interacting in a ROSCA for a short duration, there is no attempt to steal the pot when allocated and therefore agents, on the whole, behave honestly.

2. Simulation 2 - Given a mid-sized ROSCA circle with a duration of 8 months, we find an agent cheats and steals the pot, leading to spikes in agents behaving dishonestly.

3. Simulation 3 - With the largest number of participants and longest duration of a year (12 months), we see an there's a 50per-cent chance of an agent stealing the pot, if so, as seen before it leads to spikes in agents behaving dishonestly, if not the dishonest behaviour

throughout the duration is negligible.

From the results above we can conclude, if we aim to minimise dishonest behaviour within our system, we could set rules that dictate the size of a circle an agent can enter according to their trust rating, for an example: A new agent that has no trust rating can accrue a rating by entering a smaller sized ROSCA where agents are unlikely to steal the pot.

It is important to note the more participants there are in a ROSCA, the higher the sum collected at the end of the month. Therefore, an agent with a cheating strategy may utilise this information and enter a circle with a high contribution amount, with the intent to steal. We can test if there is a correlation between total groups pool amount and whether agents are more or less likely to cheat in Analysis 4.

5.1.4 Analysis 4

How does an increase in amount agents are allowed to enter with impact the outcomes of the ROSCA circle?

For an example:
A lower maximum amount may lead to less or no stealing the pot as the reward isn't much higher than the cost of stealing. eg: a circle may define the maximum monthly contribution amount as 200, therefore, a person can contribute is 200 per month. Giving them a total of 200 * 10 months = £2000 when it is their turn to be allocated the pot.

To conduct this analysis, we need to simulate three different versions of the same system, by sweeping three values for Maximum Monthly amount, namely:

1. ROSCA circle with 10 agents with a duration of 10 months and a maximum monthly amount of 200.
2. ROSCA circle with 10 agents with a duration of 10 months and a maximum monthly amount of 400.
3. ROSCA circle with 10 agents with a duration of 10 months and a maximum monthly amount of 60.

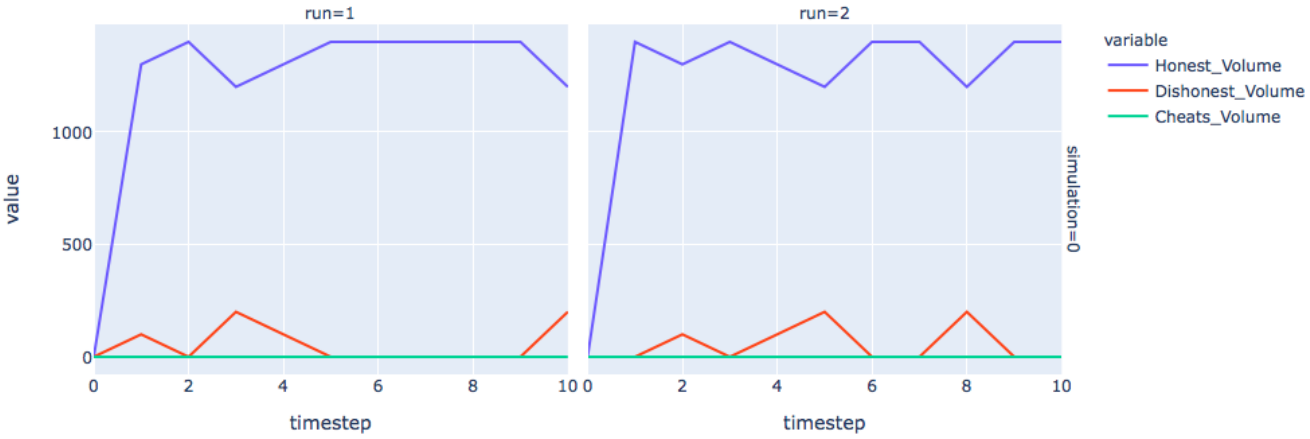

Figure 5.6: Simulation with parameter max amount at 200

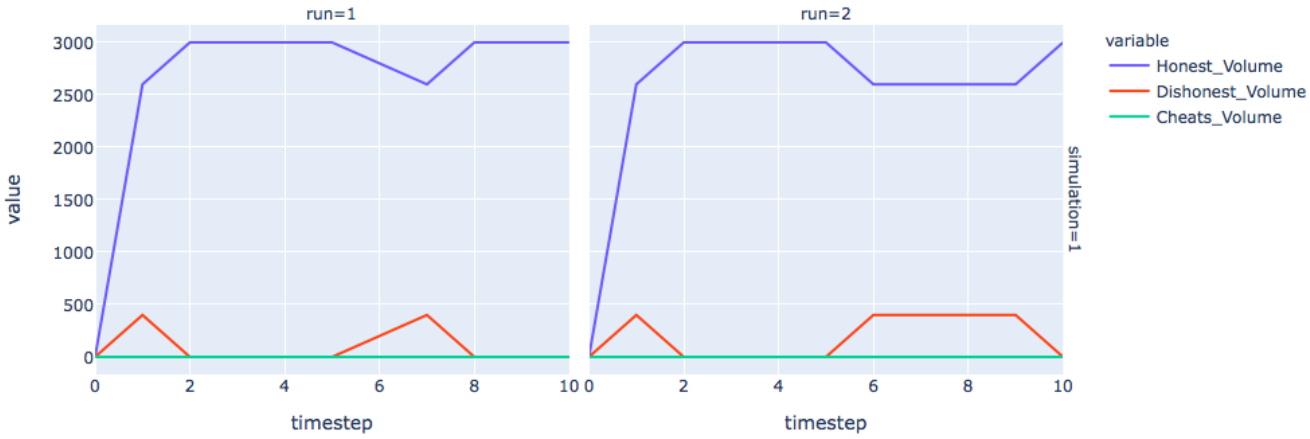

Figure 5.7: Simulation with parameter max amount at 400

Figure 5.8: Simulation with parameter max amount at 600

Experiment Results

We run two Monte-Carlo runs for each simulation to obtain a second set of results for each parameter sweep. From the three simulations above we note a few things:

We find there is a correlation between the amount agents are allowed to enter in the ROSCA and whether or not agents are then likely to Cheat and steal the pot. Only at the highest maximum amount of 600GP/month does an agent cheat.

However, it is also important to point out, since both runs of each simulation start with the same initial state (same agents), the same agent with a high-risk attitude steals the pot (at timestep 1) in both runs. Knowing this, we can say whether an agent cheats or not, for the most part, depends on their attitude to risk and their ROI.

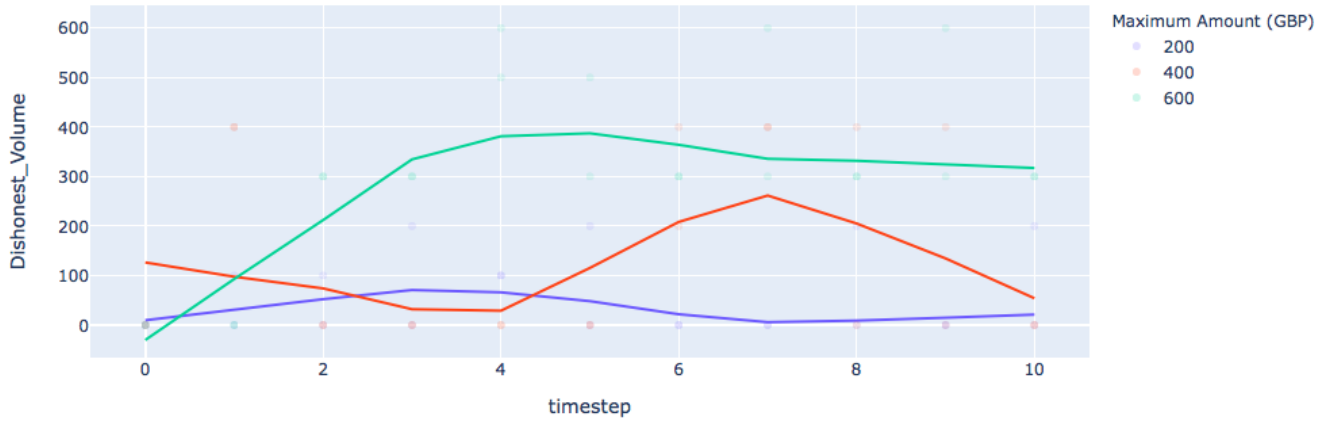

Figure 5.9: Variations of Maximum Amount and Dishonest Volume

In the figure above, we analyse whether there is a relation between dishonest behaviour and the maximum amount allowed. We see a trend of increasing dishonest Volume at increasing Maximum Amounts.

5.2 Critical Analysis

In this section, we validate whether or not our model has met our requirements set out (3.2.2).

1. Present the dynamics of a single ROSCA circle on a peer-to-peer decentralised network where agents share a common view of the chain.

The finished program simulates the dynamics of ROSCA on a blockchain network. In creating each ROSCA circle, we first initialise our genesis block containing data outlined in 5.2. As agents interact with each other and the system, this data is recorded onto subsequent blocks which form the blockchain. This serves two practical purposes:

1. During the duration of a given ROSCA, it allows for system decisions to be made based on agents actions which are stored on the blockchain. For example the decision of whether or not to allow an agent to borrow.
2. At the completion of a ROSCA this record of immutable data is available.

Since our model is a simulation, there exists only a single node on the network, however, in practice, each agent acts as a node fulfilling our aim of a distributed network.

2. Define simple strategies for agents to join a ROSCA circle (with either known or unknown peers) and receive and put money into the pot

In our model, we have defined functions for agents to join a given ROSCA. These agents can be either who know one another or not. We have defined two core policy functions, one to initiate monthly contributions from agents in the circle (3a) and another to allocate it to an agent (3b).

3. Agents can choose their contribution amount and the groups duration

In simulating the final component of our system, we create a function that allows agents to choose their contribution amount (2). For our final system analysis, we use the duration of a ROSCA as a system parameter, simulating shorter and longer ROSCAs (5.1.3) to answer questions about our model.

4. Agents have a 'trust' rating, a metric based on a participants' actions, which is built over time.

This is essentially our incentive system. We create this in the second component of our system, the Agent-Based Model (4.3). Agents have a personal rating, which increases as they behave honestly, contribute each month and continue to contribute after they receive their allocation.

We see this system get put to use in ordering the turn in which agents are allocated the pot and be taken into consideration when an agent proposes to borrow from the pool (3).

5. Agents are capable of utilising different strategies to maximise their reward whilst minimising their cost.

The basis of building an Agent-Based Model was defining strategies for agents which determined how they behaved within a ROSCA. We defined three types of agent contributor strategies (1) and two strategies for agents to whom the pot is allocated (1). At each point each agent makes a decision about how to act, this is made by weighing functions (3) that allow agents to calculate their reward and cost.

6. Agents can define rules governing their ROSCA and reach consensus with other agents to approve or disapprove borrowing

We introduced a simple model for governance in 4.4, with a mechanism allowing for a participant vote allowing participants to reject or approve proposals such as borrowing. However, we did not meet our requirement of agents defining their own rules, in addition to the implemented system rules governing the ROSCA.

Chapter 6

On Reflection

6.1 Conclusions

Our overarching aim was to prove that ROSCAs could operate without established trust or a central authority figure. We went further and asked whether we could prove this in addition to improving the current model of ROSCA.

We proposed a model which utilised the inherent trust-less feature of blockchain, which essentially allows a group of like-minded people who do not know each other to form a community in pursuit of their financial goals. In designing our protocol we, incorporated mechanisms to reflect keywords from our key question: 'Is my proposed model of ROSCA more **fair**, **democratic**, **decentralised** than existing systems?'

1. Fair - We implemented Smart Contracts to dictate the flow of funds, code is inherently unbiased, whilst ensuring that all transactions comply with the underlying legal agreements.

2. Democratic - We implemented a mechanism for voting, allowing participants to coordinate and make collective decisions.

3. Decentralised - We have successfully designed our system to operate without a central authority, with decisions being made by each participant. In addition to this, the ROSCA circles ledger of transactions and agent actions, is shared across all nodes of our distributed network.

From our system analysis, we aimed to deduce how well our system as a whole performed to

facilitate our goal (1.1). We found through carefully designed incentives, where agents are free to pursue their distinct strategies, on the whole, they behave in a way that fulfils the goals of the protocol. From this we can conclude, having met all our aims and objectives, we have successfully demonstrated we have significantly improved on the current models of ROSCA and that our proposed model is in fact more "fair, democratic, decentralised than existing systems".

Having said that, there are still further aspects that could be developed to push the project further as discussed in section 7.2.

6.2 Model Improvements

6.2.1 Where did the proposed model fall short?

- We originally intended to simulate two types of ROSCA models: one where participants have established trust and one without. However, due to falling behind on time, we opted to solely focus on modelling the latter to meet our aims. In terms of analysis, it would have been interesting to use a variation of our model where agents are highly unlikely to cheat (established trust) and compared it to our model to determine how well our designed incentives worked.

- We would have also have liked to analyse the incentives and governance system by answering the questions: Does the incentive system serve its purpose in prioritising honest agents over dishonest agents whilst also influencing behaviours of agents due to associated rewards? And what effect does the addition of governance have on agent behaviours, if any?

- Another feature we also intended to include was to allow agents to reach consensus on members of their circle before the ROSCA process commenced. This would have served as a penalty for agents with a history of dishonest behaviour.

- In terms of how we could facilitate a system of governance, we created a simple mechanism for voting which was based on system rules and a participant vote that could either support the decision or override and reject it. We would have liked to implement a more sophisticated form of voting, using the Proof of Virtual Voting (PoVV) consensus protocol. In our voting system, there is the possibility of malicious nodes influencing others' votes or preventing some from participating, this can be avoided using PoVV, which assigns

votes to agents based on mathematical calculations. We could apply this to our model so that the network assigns positive votes to an agent if they carry on contributing after receiving a large allocation of funds or paying back funds after borrowing. These votes could serve to reinforce or replace our incentive system (trust-rating).

6.2.2 Future work

There are a few opportunities for extending this model:

Extention 1:

In this model, we have proposed a basic governance model, where we defined basic rules governing the ROSCA. We could extend this to allow agents to include additional rules for their specific ROSCA.

Extention 2:

What role can monetary incentives play in our system? eg: agents that behave honestly with a corresponding high trust rating could be rewarded with an additional bonus.

Extention 3:

Can we use machine learning to predict which agents are likely to cheat, given we can use an input of their history and past behaviours?

6.3 Socio-economic and academic impact

What is the significance of our model?

Our proposed model of decentralised ROSCA contributes to the wider Decentralized finance (DeFi) ecosystem. DeFi is an open and global financial system – an alternative to a system that's opaque, tightly controlled, and held together by decades-old infrastructure and processes.[30] At our most ambitious we have identified, validated and tested a use-case for DeFi, more specifically, the booming lending/borrowing crypto-economy which serves to vastly improve financial inclusion.

In conducting our literature review we found a startup, We Trust, currently developing a blockchain-based platform for ROSCAs[18]. We, therefore, believe while in their phase of development they stand to gain practical insights from our model which has had its assumptions, mechanisms and system behaviours validated by simulation.

6.4 Personal Development

This project has allowed me to challenge myself significantly whilst both building and expanding my knowledge in designing and simulating decentralised economic systems.

The biggest challenge initially, was understanding how one went about engineering decentralised economic systems. In conducting a search for academic papers on the subject, I learnt how to dissect key information from high-level writing, and understand how these ideas could then be implemented in practice to a personal finance system.

This search for a wider understanding of the topic lead me to gain an overall understanding of game theory, designing incentives, the method of engineering distributed/decentralised systems and designing blockchain-based systems. Another significant area I've developed is being able to analyse results from a simulation and draw valid conclusions, whilst objectively appraising a system as a whole.

Another challenge was learning and understanding the method of token engineering and cadCAD which formed the basis of my project. Initially I began with learning the fundamentals of the python language itself and from there diving into understanding the relatively new python library, cadCAD. I found it difficult at times to find documentation on aspects of cadCAD, and while the community guides were a good starting point, I found I ended up learning a lot through trial and error.

Chapter 7

Bibliography and Citations

[1] Gonzalez, Laura, Blockchain, herding and trust in peer-to-peer lending, Managerial finance, 2019-05-13, Vol.46 (6), p.815-831

[2] Rupert Jones, Peer-to-peer lending: 'I'm 19,050th in the queue to get my savings back'. [Online]. Available: https://www.theguardian.com/money/2020/oct/17/peer-to-peer-lending-savings-covid-pandemic

[3] Crosman, Penny, Can blockchain technology revive peer-to-peer lending. [Online]. Available: https://www.americanbanker.com/news/can-blockchain-technology-revive-peer-to-peer-lending,

[4] P. Belleflamme, T. Lambert, and A. Schwienbacher, "Individual crowd- funding practices," Venture Capital, vol. 15, no. 4, pp. 313–333, 2013.

[5] P. Belleflamme, T. Lambert, and A. Schwienbacher, "Crowdfunding: Tapping the right crowd," J. Bus. Venturing, vol. 29, no. 5, pp. 585–609, 2014

[6] L.Micallef, P.Rodgers,"EulerAPE:Drawingarea-proportional3-Venn diagrams using ellipses," PloS One, vol. 9, no. 7, 2014, Art. no. e101717.

[7] Zhu, H., Zhou, Z.Z. Analysis and outlook of applications of blockchain technology to equity crowdfunding in China. Financ Innov 2, 29 (2016). https://doi.org/10.1186/s40854-016-0044-7

[8] Bogusz, Claire Ingram; Laurell, Christofer; Sandstrom, Christian, Tracking the Digital Evolution of Entrepreneurial Finance: The Interplay Between Crowdfunding, Blockchain Technologies, Cryptocurrencies, and Initial Coin Offerings, 2020-11, Vol.67 (4), p.1099-1108

[8] Sean Stein Smith, Blockchain's potential to revolutionize crowdfunding. September 23, 2019. [Online] Available: https://www.ibm.com/blogs/blockchain/2019/09/blockchains-potential-to-revolutionize-crowdfunding/

[9] Vasilios Filip, 7 Use Cases for Blockchain-based Crowdfunding, September 23, 2019. [Online] Available: https://www.datadriveninvestor.com/2019/09/23/possible-use-cases-for-blockchain-based-crowdfunding/

[10] Abbi M. Kedir, Richard Disney, Indraneel Dasgupta, Why use ROSCAs when you can use banks? Theory, and evidence from Ethiopia, 29 October 2012,[Online] Available: https://www.dse.univr.it/documenti/Seminario/documenti/documenti098832.pdf

[11] WeTrust Editorial Team, We Should Use Blockchain for Savings, Credit, and Insurance Independence, Jan 29, 2017. [Online] Available: https://blog.wetrust.io/we-should-use-blockchain-for-savings-credit-and-insurance-independence-84b3ae8fcb23

[12] P. Treleaven, R. Gendal Brown and D. Yang, "Blockchain Technology in Finance," in Computer, vol. 50, no. 9, pp. 14-17, 2017, doi: 10.1109/MC.2017.3571047.

[12] Nakamoto, Shatoshi. Bitcoin: A Peer-to-Peer Electronic Cash System. [Online] Available: https://bitcoin.org/bitcoin.pdf

[13] Glaser, Florian and Bezzenberger, Luis, "Beyond Cryptocurrencies - A Taxonomy of Decentralized Consensus Systems" (2015). https://aisel.aisnet.org/ecis2015$_cr$/57

[14] http://www.collaborativefinance.org/community/saving-circles1.png

[15] Bogusz, Claire Ingram ; Laurell, Christofer ; Sandstrom, Christian. Tracking the Digital Evolution of Entrepreneurial Finance: The Interplay Between Crowdfunding, Blockchain Technologies, Cryptocurrencies, and Initial Coin Offerings. 2020-11, Vol.67 (4), p.1099-1108

[16] Token engineering community. So, what is "token engineering?" [Online] Available: https://tokenengineeringcommunity.github.io/website/docs/getting-started-welcome

[17] TwoSolitudes "The Joys of Pretending to Help the Poor: The Kiva Story". D ailykos (2014).

[18] WeTrust whitepaper,[online] https://github.com/WeTrustPlatform/documents/blob/master/WeTrustWhitePaper.pdf

[19] Shermin Voshmgir and Michael Zargham, Foundations of Crypto-economic Systems, Nov 2019. [Online] Available: https://epub.wu.ac.at/7309/8/Foundations20of20Cryptoeconomic20Systems.pdf

[20] Brandon Ramirez. Modelling Crypto-economic Protocols as Complex Systems, Jan 14 2020. [Online] Available: https://thegraph.com/blog/modeling-cryptoeconomic-protocols-as-complex-systems-part-1

[21] Stephan Young. A token engineering process. [Online] Available: https://syoung.org/2019/10/18/a-token-engineering-process/

[22] Micheal Zargham, Introduction to Systems Thinking, Sep 2019 [Online] Available: https://community.cadcad.org/t/introduction-to-systems-thinking/18

[23] An open-source Python package that assists in the processes of designing, testing and validating complex systems through simulation. [Online] Available: https://cadcad.org

[24] Yang, Z., Lang, W., Tan, Y. (2005). Fair micropayment system based on hash chains. Tsinghua Science and Technology, 10(June (3)), 328–333.

[25] Florian Prange, On playing non-equilibrium games: Modelling the complex dynamics of games, Ecological Complexity, Volume 3, Issue 4, 2006, Pages 302-306,

[26] Wikipedia, Differntial Games, [Online] Available: https://en.wikipedia.org/wiki/Differential_game

[27] Huang, J., Kong, L., Chen, G., Wu, M., Liu, X., Zeng, P. (2019). Towards secure industrial iot: Blockchain system with credit-based consensus mechanism. IEEE Transactions on Industrial Informatics, (December), 1.

[28] Shamseya, [Online] Available: https://shamseya.org/ideas-lab/rotating-savings-and-credits-associations-roscas/

[29] Vikas Hassija, Vinay Chamola, Sherali Zeadally BitFund: A blockchain-based crowd funding platform for future smart and T connected nation

[30] Etherum, [Online] Available: https://ethereum.org/en/defi/

[31] Chen, Dongyu ; Lai, Fujun ; Lin, Zhangxi, A trust model for online peer-to-peer lending: a lender's perspective, 2014-05-31, Vol.15 (4)

[32] Celcius, [Online] Available: https://celsius.network

[33] Bloom, [Online] Available: https://bloom.co

[34] Salt Lending, [Online] Available: https://saltlending.com

[35] Michael Zargham, Jan 2019, Non-Equalibrium Dynamics. [Online] Available: https://medium.com/block-science/hi-b26f215dc6f8

[36] Wikipedia, Monte Carlo method. [Online] Available: https://en.wikipedia.org/wiki/Monte$_C arlo_m ethod$

[37] Timothy Andrew, Monte Carlo Method. [Online] Available: https://timothyandrew.dev/learning/wiki/mat carlo/