Cape Peninsula
University of Technology

PROGRAMMABLE MONEY FOR TRUSTWORTHY CURRENCY TRANSACTIONS

by

NTETHELELO LUNGELO NDABA

Thesis submitted in fulfilment of the requirements for the degree

Master of Technology: Information Technology

in the Faculty of Informatics and Design

at the Cape Peninsula University of Technology

Supervisor: Boniface Kabaso

Cape Town

December 2022

DECLARATION

I, Ntethelelo Lungelo Ndaba, declare that the contents of this dissertation/thesis represent my own unaided work, and that the dissertation/thesis has not previously been submitted for academic examination towards any qualification. Furthermore, it represents my own opinions and not necessarily those of the Cape Peninsula University of Technology.

Signed

07 December 2022

Date

ABSTRACT

With bitcoin being the first blockchain success story, followed by other cryptocurrencies such as Ethereum, there has been a keen interest in potential technologies on the blockchain platform from different sectors, including finance, education, and supply chain. Conditional payments are an example of potential solutions being investigated. Programmable money, a new trustworthy virtual currency, provides for conditional payments. It is a scenario where the behaviour is removed from the human and embedded in the money using attaching contracts to the currency itself.

This research follows the design science research (DSR) methodology to develop and evaluate programmable money by pivoting the capabilities of smart contract technology.

Using a simulation, the artefact is subjected to realistic conditions to determine the feasibility of the programmable money concept in terms of functional and non-functional performance. Quantitative data generated during the observation of the simulation was used to obtain the findings.

The findings show a strong performance in the functional test where the evaluation criteria focused on how the artefact addresses contract breaches attempted by simulation users.

The non-functional tests were centred around performance under stress to gauge how the artefact would perform when processing a) many requests while the number of concurrent users remained constant and b) many requests caused by the periodically increased number of concurrent users.

In the concurrency simulation, the findings show a quadratic time complexity $\mathrm{T(n)} = \mathrm{O(n^2)}$ compared to a linear time complexity function of $\mathrm{T(n)} = \mathrm{O(n)}$ observed where the concurrency level remained the constant.

Keywords: Programmable money, conditional payments, smart contracts, blockchain, virtual currencies

ACKNOWLEDGEMENTS

I wish to thank:

- My Heavenly father, first and foremost.
- My parents Nonsikelelo Gwiji and Mbuyiselwa Ndaba for raising me.
- My siblings, Zama and S'boniso Ndaba, for unconditional love and support.
- My son Zalumabo Ndaba and his mother Abulele Xwelesha for giving me a whole new purpose.
- Dr. Boniface Kabaso for his support throughout this endeavour.

DEDICATION

This is dedicated to my mother, Nonsikelelo Rejoice Gwiji. She is why this is possible, and I am nothing without her.

TABLE OF CONTENTS

DECLARATION 2

ABSTRACT 3

ACKNOWLEDGEMENTS 4

DEDICATION 5

TABLE OF CONTENTS 6

LIST OF FIGURES 13

LIST OF TABLES 15

GLOSSARY 16

CHAPTER ONE : INTRODUCTION TO THE RESEARCH STUDY 18

1.1 Introduction 18

1.2 Background to the research problem 18

1.2.1 Blockchain and trust 18

1.2.2 Programmable Money 19

1.3 Research Problem 20

1.4 Aim and Objectives 20

1.5 Research Questions 21

1.6 Research Methodology 21

1.7 Limitation of the study 22

1.8 Outcomes, Contribution, Significance 23

1.8.1 Outcomes 23

1.8.2 Contributions 23

1.8.3 Significance 23

1.9 Thesis Outline 24

CHAPTER TWO : LITERATURE REVIEW 26

2.1 Blockchain Overview 26
2.1.1 Integration of blockchain with supply chain 26
2.1.2 Consensus Algorithms 27
2.2 Programmable Money 28
2.3 Automation 31
2.3.1 IFTTT (If This Then That) 31
2.3.2 Smart Contracts 32
2.4 Systematic Literature Review 33
2.4.1 Systematic Literature Review Questions 34
2.4.2 Systematic Literature Review Protocol 35
2.4.2.1 Search Strategy 35
2.4.2.2 Search terms 36
2.4.2.3 Search results 37
2.4.2.4 Study Selection 37
2.4.2.5 Quality Assessment 38
2.4.3 Search Results 39
2.4.4 Results of the review 41
2.5 Chapter Summary 44
CHAPTER THREE : RESEARCH METHODOLOGY 46
3.1 Introduction 46
3.2 Research Pyramid 46
3.2.1 Research Paradigm 47
3.2.2 Research Methodology 48
3.2.3 Research Methods 49
3.2.4 Research Techniques 50
3.3 Design Science Research (DSR) 51

3.3.1 Application of the guidelines to the proposed artefact 51
3.3.1.1 Design as an artefact 52
3.3.1.2 Problem relevance 52
3.3.1.3 Design Evaluation 52
3.3.1.4 Research Contributions 54
3.3.1.5 Research Rigor 55
3.3.1.6 Design as a search process 55
3.3.1.7 Communication of research 56
3.4 Data Collection 56
3.5 Data Generation and Analysis 57
3.6 Summary 58
CHAPTER FOUR : DESIGN OF PROGRAMMABLE MONEY ON ALGORAND BLOCKCHAIN 59
4.1 Introduction 59
4.2 Design Goals 59
4.3 Design Objectives/Requirements 60
4.4 System Overview 60
4.4.1 Conceptual Model 61
4.4.2 System Components 62
4.4.2.1 Tokens with dynamic conditions 62
4.4.2.2 Smart Contract 62
4.4.2.3 Atomic Transfer 62
4.4.2.4 Assets 63
4.4.2.5 Token Owner 64
4.4.2.5 Spender 64
4.5 Deployment 64

4.5.1 Virtual Machine 64
4.5.2 Server 64
4.5.3 Infrastructure 65
4.6 Resource requirements 66
4.7 Chapter Summary 67
CHAPTER FIVE : EVALUATION OF PROGRAMMABLE MONEY THROUGH SIMULATION 68
5.1 Introduction 68
5.2 Problem Domain 68
5.3 Simulation Outline 68
5.3.1 Participants 69
5.3.2 Experiment Set up 69
5.3.2.1 Stateful smart contract application 69
5.3.2.2 Escrow Account 71
5.3.2.3 Creation Script 72
5.4 Test cases 72
5.5 Functional Simulation Results 73
5.5.1 Payment to non-whitelisted account 74
5.5.2 Payment from a non-whitelisted account 75
5.5.3 Transfer assets with insufficient transaction fee 75
5.5.4 Transfer without a transaction fee 76
5.5.5 Transfer STR outside the permitted range 77
5.5.6 Test for success 78
5.6 Performance-Based Simulation Result 79
5.6.1 Repetition Simulation 79
5.6.1.1 Data transfer per transaction count 80

5.6.1.2 Transaction rate 81
5.6.1.3 Throughput 81
5.6.2 Concurrency Simulation 82
5.6.2.1 Data transfer per batch 82
5.6.2.2 Transaction rate per batch 83
5.6.2.3 Throughput 84
5.6.3 Multi URL Simulation 84
5.6.3.1 Data transferred per round 85
5.6.3.2 Transaction rate 85
5.6.3.3 Throughput 86
5.7 Chapter Summary 87
CHAPTER SIX : RESEARCH FINDINGS AND DISCUSSION 88
6.1 Introduction 88
6.2 Discussion of results 88
6.2.1 Evaluation of transfers to non-whitelist accounts 88
6.2.2 Evaluation of transfers with insufficient or non-existent transaction fees 89
6.2.3 Evaluation of transfers where the transfer amount violated the permitted range 89
6.2.4 Evaluation of transactions that do not violate agreements 89
6.2.5 Performance evaluation 89
6.2.5.1 Time elapsed comparison 90
6.2.5.2 Concurrency 90
6.2.5.3 Time elapsed as the number of requests increases 91
6.2.5.4 Time elapsed as the number of requests increases in multiple URL simulation 92

6.3 Problems encountered and Provided Remedies 92
6.3.1 Deployment on long-running networks 92
6.3.2 Creating dynamic participants during runtime 93
6.3.3 Use of cron jobs 93
6.3.4 Transaction times 93
6.3.5 Zero value transfers and negative transfers with charges 93
6.3.6 Feedback on exception handling 94
6.3.7 Service endpoints 94
6.3.8 Evaluation of longevity 94
6.4 Research Findings 94
CHAPTER SEVEN: RESEARCH EVALUATION AND CONCLUSION 98
7.1 Hevner's DSR Guidelines 98
7.1.1 Design as an artefact 98
7.1.2 Problem relevance 98
7.1.3 Design Evaluation 99
7.1.4 Research Contributions 99
7.1.5 Research Rigor 100
7.1.6 Design as a Search Process 100
7.1.7 Communication of Research 101
7.2 Conclusion 101
7.3 Future Work 103
REFERENCES 104
APPENDICES 110
APPENDIX A: Create Smart Contract Application and Token bash script 110
APPENDIX B: Add whitelist account bash script 111
APPENDIX C: Transfer Bash Script 112

APPENDIX D: Extract of permission smart contract validations 113
APPENDIX E: Transfer from non-whitelisted account test bash script 114
APPENDIX F: Transfer to non-whitelisted account test bash script 115
APPENDIX G: Transfer outside of allowed asset amount range test bash script 116
APPENDIX H: Transfer without payment to escrow test bash script 117

LIST OF FIGURES

Figure 1.1: Design Science Research Process Model 22

Figure 2.1: The core functionality of a campaign contract 30

Figure 2.2: Monzo + IFTTT Trigger actions 32

Figure 2.3: Systematic guide to literature review development 34

Figure 3.1 The Research Pyramid 47

Figure 3.2: Design science research process (DSRP) model 49

Figure 3.3: Scaling up the conditions of practice 53

Figure 3.4: Example scenario of scaling up to more realistic conditions 54

Figure 3.5: Software engineering methodologies 55

Figure 3.6: Research process followed 57

Figure 4.1: Abstract Representation of the Programmable Money solution 61

Figure 4.2: Automatic Transfer Flow 63

Figure 4.3: BetaNet, TestNet, and MainNet 65

Figure 4.4: Infrastructure where the artefact is deployed 66

Figure 5.1: Transfer Transaction 71

Figure 5.2: Payment to non-whitelisted accounts 74

Figure 5.3: Payment from a non-whitelisted account 75

Figure 5.4: Transfer assets with insufficient transaction fee. 76

Figure 5.5: Transfer transactions are not in violation of the contract 79

Figure 5.6: Data transferred per batch 80

Figure 5.7: Transaction rate per batch 81

Figure 5.8: Throughput per batch 82

Figure 5.9: Data transferred per batch 83

Figure 5.10: Concurrency simulation – Transaction rate per batch 83

Figure 5.11: Concurrency simulation - Throughput per batch 84
Figure 5.12: MultiURL - Data transferred per round 85
Figure 5.13: Multi-URL – Transaction rate per batch..... 86
Figure 5.14: Multi-URL - Throughput per round 86
Figure 6.1: Difference in the time elapsed for each test in the performance simulation 90
Figure 6.2: Elapsed time per round in the concurrency simulation..... 91
Figure 6.3: Elapsed time per round in the multiple-request simulation..... 91
Figure 6.4: Elapsed time per round in the multiple URL’s simulation 92

LIST OF TABLES

Table 2.1: Systematic Literature Review Questions 35
Table 2.2: Online search databases 35
Table 2.3: Search terms 37
Table 2.4: Search results 37
Table 2.5: Inclusion criteria 38
Table 2.6: Exclusion Criteria 38
Table 2.7: Quality Assessment Criteria 39
Table 2.8: Refined results 39
Table 2.9: Quality Assessment of Studies 41
Table 2.10: Comparison of the different solutions 43
Table 3.1: Mapping DSR activities to current research 49
Table 3.2: DSR Guidelines 51
Table 4.1: Resource requirements 66
Table 5.2: Account metadata 74
Table 5.3: Transfer assets without a transaction fee 77
Table 5.4: Transfer of STR in outside of the permitted range 78

GLOSSARY

Terms/Acronyms/Abbreviations	Definition/Explanation
ASA	Algorand Standard Asset
AVM	Algorand Virtual Machine
BFT	Byzantine Fault Tolerance
DR	Design Requirement
DSR	Design Science Research
DSRM	Design Science Research Methodology
DSRPS	Design Science Research Process
IFTTT	If This Then That
MultiSig Account	Multiple signature account
PBFT	Practical Byzantine Fault Tolerance
PoA	Proof of Authority
PoS	Proof of Stake
PoW	Proof of Work
RBFT	Redundant Practical Byzantine Fault Tolerance
RQ	Research Question
SC	System Component
SCF	Supply Chain Finance
SLR	Systematic Literature Review

SLRQ	Systematic Literature Review Question
STR	Stable Rand
STRC	Stable Rand Coin
VM	Virtual Machine

CHAPTER ONE : INTRODUCTION TO THE RESEARCH STUDY

1.1 Introduction

With the buzz surrounding bitcoin having reached fever pitch, many authors claim blockchain to be the real innovation. Different authors list multiple technologies that have the potential to be introduced on the blockchain platform to improve current systems. They include digital certificates, Escrow services for goods, systems designed to combat electoral fraud, transfer of deeds or bonds, voting systems, and supply chain management (Tsilidou and Foroglou, 2015; White, 2017).

Kiviat (2015) simply defines blockchain as a solution to an elusive network problem by enabling 'trustless' transactions where central authorities are no longer needed to monitor, verify and enforce value exchanges over computer networks.

Although conventional transactions are often centralized and monitored through an additional instance, blockchain technology is popular as an enabling technology due to its ability to establish trust between participants in a decentralized network without needing a third party (Omran *et al.,* 2018).

Yaga *et al.* (2018) define blockchain as a tamper-proof digital ledger implemented decentralized (without a central repository) and typically without a central authority. Each transaction in the public ledger is immutable and is also verified through majority consensus.

Cosby et al. (2016) use an interesting analogy to motivate the fundamental benefits of blockchain. "It is easier to steal a cookie from a cookie jar that's kept in a secluded place, than stealing a cookie from a cookie jar kept in a marketplace, being observed by thousands of people". The analogy emphasizes the power of consensus, an important property of blockchains.

1.2 Background to the research problem

1.2.1 Blockchain and trust

A verification at notaries or a registration through intermediaries such as banks is not only necessary but inevitable when engaging in financial transactions. This is the

current situation. Consequently, the requirement of a third party when entities engage in transactions is usually costly and time-consuming and represents a single and central point of failure (Omran *et al.* 2018). These challenges are significantly increased where supply chains are concerned about connecting multiple transactions at different levels throughout the supply chain.

Existing payment methods rely on central authorities such as banks to manage and store transactions (Wang et al., 2019). This institutional model of trust presents a range of issues, including lacking sustainability in terms of cost and time. Since transactions come with banking costs, each transaction is as fast as these authorities. It is prone to fraud and corruption and gives too much power to these central authorities (Agarwal et al., 2018; Cunha et al., 2021; Wang et al., 2019; Yaga et al., 2018).

1.2.2 Programmable Money

Programmable Money (PM), a new cryptocurrency powered by blockchain infrastructure, promises to remedy this in a highly secure platform with a decentralized trust model system where trusted third parties are eliminated. PM can mediate without the need for organizations or constitutions through the trust of a public consensus. In this system, we need not trust intermediaries or each other, but we can build trust through transparent systems that are completely neutral and have thousands of nodes that verify that each transaction is legitimate.

Recent studies on potential blockchain technologies have revealed the potential for money to be more than just a medium of financial exchange. Gladden (2015:85) argues that through artificial intelligence, machine ethics, and cryptocurrency, it is possible to invent cryptocurrencies capable of autonomously regulating their own use in a manner that mirrors the ethical values of human beings.

Elsden et al. (2019:10) present a study on a platform where a banking app integrates with a web service called If This Then That (IFTTT), which empowers money with conditional automation and enforces consequences where necessary on transactions.

These solutions attempt to address fraud and corruption in financial transactions (Pandey et al., 2019; Wang et al., 2019). An example is online crowdfunding, with only

the project manager and not the contributors having control or visibility of their donations. This creates opportunities for malicious activities (Agarwal et al., 2018; Pandey et al., 2019).

1.3 Research Problem

The problem addressed by this study deals with contract breaches related to the financial exchange of money that does not conform to the intended purpose.

This problem currently exists due to a lack of information systems dedicated to managing the flow of information between parties involved in financial transactions to reliably verify that funds are being used for their intended purposes (Hyvärinen et al. 2017:2).

This has resulted in multiple financial contracts being breached without detection. An example would be a government-allocated college fund used to buy a car or charity funds from community donations not being used for purposes that benefit the needy.

And even in places where transactions can be tracked, it is costly to have an army of auditors that take too long to produce evidence of maladministration of funds. This evidence is often contested in lengthy and costly legal battles.

It is, therefore, imperative to develop and empirically test a system that not only enforces the intended purpose use for money but also tracks the usage without the possibility of any disputes from implicated parties.

1.4 Aim and Objectives

This research aims to develop PM as a trustworthy virtual currency that autonomously enforces hardwired terms and conditions with each transaction performed.

Aligned with the aim, the key objectives of this research are as follows:

- Determine the appropriate blockchain to use as a platform.
- Create PM based on crypto tokens on the chosen platform.
- Enhance the PM based on smart contracts with enforceable rules
- Evaluate the efficiency in the enforcement of these rules mentioned above.

1.5 Research Questions

1. How can we build PM with characteristics that enable it to enforce usage based on the intent of the allocation?
    I. What are the available blockchain platforms, and how do they compare to each other?
    II. How can we create PM tokens on the chosen platform?
    III. How can we use smart contract technology to enhance the created PM tokens with enforceable rules?
    IV. What are the efficiencies and efficiency levels required to determine how well the rules are being enforced?

1.6 Research Methodology

This research will adopt the design and creation strategy to design an artefact. According to Oates (2006), design and creation is a strategy used in design science research (DSR). It is a strategy adopted when the research "requires producing a new element of a system or a system as a whole".

Hevner and Chatterjee (2010:5) DSR as a paradigm in which designers answer questions related to human problems by creating innovative artefacts, thus adding knowledge to scientific evidence. The artefact is useful and contributes to understanding the problem. This is supported by Vaishnavi et al. (2019) with their concept of gaining knowledge, which means that the researcher learns by creating or building an artefact.

DSR was found to be the appropriate paradigm for this study and is discussed in detail in Section 3.3. Section 3.3.1 describes in detail the DSR guidelines applied in this research, and the application of these guidelines is presented in Section 3.3.2.

The following is a graphical display of the process model adopted by this study.

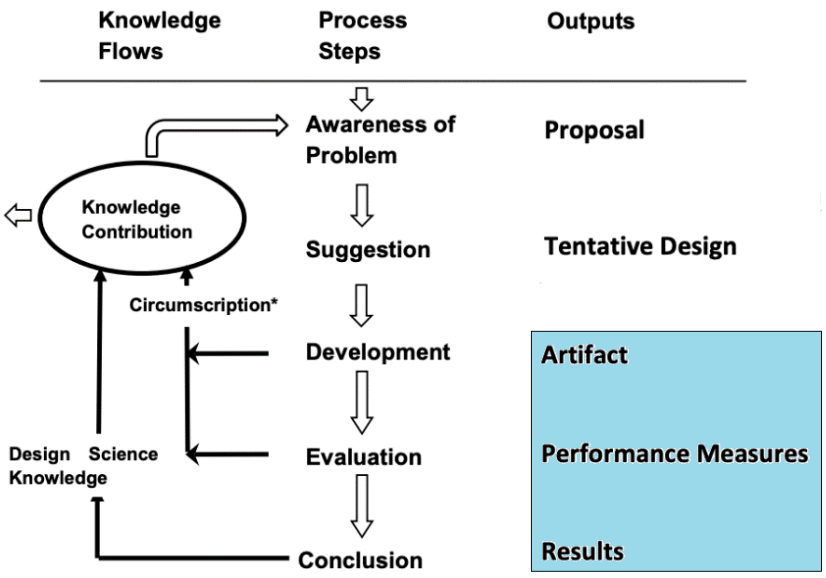

Figure 1.1: Design Science Research Process Model

(Adapted from Vaishnavi *et al.* 2019)

The artefact (PM) is proposed to address an identified gap for solutions that enforce contracts during the execution of transactions in real-time and also provide an audit trail for the flow of money.

This will be a small-scale deployment for private testing by a small research team. Algorand was chosen as the appropriate blockchain platform for deployment (Sections 2.5 and 4.5).

The developed artefact will be validated in the context of the problem through a simulation in a controlled environment. The validation was done through a simulation consisting of functional and non-functional testing (Section 5.5 and Section 5.6).

Data about the performance of the artefact was stored during the simulation and analysed quantitatively to determine the research findings discussed in section 6.4.

1.7 Limitation of the study

The contribution of this research should be considered in the light of its limitations, which are also the basis for future research.

First, the governance issues regarding law and order regarding the regulation of entities such as smart contracts are reserved for future use and will not be covered in

this study. This research is limited to empirically testing the feasibility of programmable money as a trustworthy virtual currency.

Secondly, the proposed artefact will be developed through open-source technology. Thus, as mentioned in section 1.8.2, the artefact will also be open to the community.

1.8 Outcomes, Contribution, Significance

1.8.1 Outcomes

The research conducted will produce empirical data on the problem of enforcing a smart contract. The second outcome is the proposed solution to be produced by the research. Finally, there are the thesis reports and the articles that report on the research.

1.8.2 Contributions

This research will make the following contributions:

- Empirical data about testing PM in a controlled environment.
- Data about challenges and best practices required to deploy PM.
- Data about the limitations of PM and what is required for a large-scale deployment, including performance metrics of the various methods to be used and tested.
- The methodology used in the production of the proposed artefact.
- A new library to write long-running contracts on the Algorand blockchain.

1.8.3 Significance

Through the development of PM, this research will demonstrate the feasibility of a self-regulating currency capable of combating the abuse of money currently seen in financial transactions. The artefact presents additional benefits inherited from the blockchain platform, including eliminating the intermediation layer (and the associated costs and time) when transacting.

1.9 Thesis Outline

The thesis comprises seven chapters. The thesis is introduced in Chapter 1. The background of the research, the aims and objectives, and the research problem are described.

Chapter 2: Literature Review

In the second chapter, a systematic literature review is conducted. A systematic literature review is different from other literature reviews in that it addresses the issue of selection bias, where the authors only include studies that are consistent with their individual opinions.

This chapter describes relevant discussions in the knowledge base related to the study being conducted. We will see the timeline of money leading to programmable money. This dates to the barter trade system, to fiat money, to digital money, all the way to cryptocurrencies, where transactions are capable of making autonomous decisions through artificial intelligence.

Chapter 3: Research Methodology

This chapter discusses design science research as the recipe chosen for executing the research. The research methodology is structured around the research pyramid broken down into four interconnected levels: the research paradigm (at the highest level), research methodology, research methods, and research techniques (at the lowest level). This chapter will provide step-by-step DSR guidelines aligned with this research.

Chapter 4: Design of Programmable Money on Algorand

Against information collected through the systematic literature gathered in chapter 2, this chapter presents the design concept of this research. This design concept is constructed in line with the design science research methodology discussed in Chapter 3.

Chapter 5: Evaluation of Programmable Money through Simulation

This chapter reports on an evaluation of the artefact through simulation. In the simulation, the artefact is subjected to multiple tests that present life-like scenarios to

gauge how efficiently the artefact enforces contracts and if the artefact provides a solution to the given problem.

Chapter 6: Research Findings and Discussion

The data collected from the simulation conducted in Chapter 5 is analysed in this chapter. The research findings are mapped back to the research objectives. Problems encountered during the research are also discussed along with provided remedies.

Chapter 7: Research Evaluation and Conclusion

This chapter maps the actions taken in the research to the theoretical framework adopted in this research to evaluate adherence to the theoretical framework. The chapter uses Hevner's seven DSR guidelines and identifies work done in this research to each guideline. Future work on the topic is also discussed, and the thesis is concluded.

CHAPTER TWO : LITERATURE REVIEW

All research work should be based on existing knowledge in the subject area. A literature review identifies and organizes concepts in the relevant literature (Rowley & Slack, 2004:31).

This chapter presents the work and opinions of other authors and forms a background theory for this thesis. Sections 2.1 – 2.3 is a review of opinions related to programmable money. This includes the blockchain (Section 2.1), a platform on which the proposed artefact is developed, and then programmable money (Section 2.2) and automation (Section 2.3), which are related to money and financial intelligence. Section 2.4. presents a systematic literature review focusing on programmable money, and the chapter is summarized in section 2.5.

2.1 Blockchain Overview

Blockchain is an open distributed ledger that can efficiently and permanently record transactions between two parties in a verifiable way (Bach et al., 2018).

Singh and Singh (2016) define blockchain as the digital ledger of all cryptocurrency transactions. It keeps growing as the "completed" block is added with a new set of recordings. Each block contains a cryptographic hash of the previous block, a timestamp, and transaction information. Bitcoin nodes use the blockchain to distinguish between legitimate Bitcoin transactions and attempts to re-spend coins that have already been issued elsewhere.

2.1.1 Integration of blockchain with supply chain

A lack of focus on business-to-business integration in supply chains may be a significant factor in the prevalence of issues surrounding supply chain finance. An important part of this B2B integration is exchanging information over the internet between parties and value-added service providers in the supply chain. As a remedy, a digital supply chain (DSC) was born to create an end-to-end information system within the supply chain network (Korpela et al*.,* 2017).

In a DSC environment, companies with unique goals, including competitors, collaborate as hub organizations leading the integration work. Transactions related to process and data integration are usually executed by trusted intermediaries, most often through banks (Tseng and Shang, 2021).

Advocates of blockchain technology have promised to remove these third parties in a system that includes a public ledger of transactions while using public key infrastructure that can alert all parties in the DSR about executable transactions and the concept of the smart contract. The key question is about discovering how blockchain technology can support the digital supply chain. The aim is to implement common solutions, technology, and standards for integrating business processes within a large supply chain (Korpela *et al.,* 2017).

2.1.2 Consensus Algorithms

An important part of blockchain technology is the consensus model. This is how the nodes agree on the validity of a transaction. Consensus models provide a way for distrusting nodes to work together (Yaga *et al.*, 2018; Wang *et al.*, 2019:109443)

There are two classes of consensus algorithms, the cash-tolerant consensus and the Byzantine consensus.

Algorithms belonging to the cash-tolerant consensus class aim to guarantee atomic delivery (overall order) of messages within participating nodes in the event of a certain number of node failures. These algorithms use the concept of a view or epoch to denote a specific time, period, or event. The Byzantine consensus class of algorithms aims to reach consensus among specific nodes exhibiting Byzantine behaviour. Such Byzantine nodes are believed to be under enemy control and behave maliciously and unpredictably (Ferdous et al., 2020).

The most common consensus algorithm used in blockchain networks is the proof-of-work (PoW) algorithm. PoW requires nodes to perform a computationally intensive task to validate a block of transactions. The first node to successfully validate a block is rewarded with a certain number of tokens. PoW is a very effective consensus algorithm but also very energy-intensive (Gupta and Sadoghi, 2019). As such, there

has been much interest in developing alternative consensus algorithms that are more energy efficient.

Proof of Stake is one of the leading candidates for solving the energy demand problem in current blockchain protocols such as Bitcoin and Ethereum (Siim, 2017). In cryptocurrencies, proof of stake not only proves possession of an exact amount of currency, but it can also prove that that currency is satisfying some conditions (e.g., there is a restriction on spending it until some contract is satisfied) (Poelstra, 2014).

2.2 Programmable Money

Transactions in traditional supply chains involve a range of intermediaries throughout the supply chain. Costly and complex large IT-systems, security flaws and time-consuming processing are just a few disadvantages prevailing in supply chain operations (Omran *et al.,* 2018).

Furthermore, assets with intrinsic value, such as real money, incorporate sophisticated security marks and cannot be physically in two places simultaneously, and digital data can easily be copied or intercepted. This is why intermediaries, such as banks, must execute electronic payments (Omran *et al.,* 2018).

Blockchain technologies and the internet of things are currently offering guaranteed provenance and also introduce a digital footprint (Kim *et al.,* 2018) even in those supply chains with high levels of complexity (Armstrong 2016) through internet-aware sensors that capture the following information throughout a supply chain:

- granular and real-time data about the product
- environment characteristics
- location and timestamp information.

Blockchain databases are also decentralized, eliminating the need for third-party authority, while provenance can still be evaluated even when no one party owns all the supply chain data. Moreover, distributed shared databases using blockchain technology promise highly secure and immutable access to supply chain data (Kim et al., 2018).

A blockchain system is a distributed system that depends on consensus to determine the state of a particular piece of data between distributed nodes. Consensus

algorithms are the core components that directly dictate how such systems behave and what performance they can achieve. Decentralized consensus has been a much-studied research topic in distributed systems but has received renewed attention with the advent of the blockchain (Ferdous et al., 2020). Consensus is a distributed computing problem where nodes in a system must reach a consensus when there are process failures or misleading nodes (Bach et al., 2018).

Cryptocurrency advocates argue that the ethical dimensions of cryptocurrency are understood to be passive instruments that lack ethical values and can be vulnerable to those with bad will. Gladden (2015) argues that through AI, machine ethics, and cryptography, there is a possibility of artificially intelligent cryptocurrencies that are not ethically neutral but are capable of autonomously regulating their own use in a way that reflects human ethics or even human societies.

In the shift toward today's more cashless societies, for example, new payment infrastructures, such as mobile money, daily economic exchanges are becoming increasingly digital. For this reason, digital financial services constantly engage in drives to combine money and data in new and innovative ways. The EU has even introduced regulations to support 'Open Banking', which strips banks of sole custody of customer financial data, such as spending patterns (Elsden, 2019).

Through the popularity of blockchain technology and, most notably, smart contract technology, programmable money and conditional payments have recently become a topic of active interest in recent years (Elsden *et al.*, 2019; Pandey *et al.*; 2019; Weber and Stables, 2021). Pandey et al. (2019:1030) propose a method to manage funds in a crowdfunding campaign based on blockchain technology. Using the Ethereum blockchain, they developed smart contracts that autonomously apply predetermined conditions, such as processing a payment only after 50% of the contributors in a campaign approve the payment request.

Elsden et al. (2019:379) explore programmable donations through escrow-based conditional giving. In this platform, the donor creates a 'smart donation' that contains an '*offer'* comprising the value of the donation. The associated '*conditions'* under which the donation can be withdrawn. These conditions trigger the *validator(s)* who provide data that inform smart contracts if conditions/ agreements are being violated. Finally,

donations define a '*beneficiary'* – the individual, cause or account who would receive the funds and an expiry date when any unreleased funds are returned to the donor.

Weber and Staples (2021:1-2) embark on a 'Smart Money' project to explore systems for programmable money on the Ethereum blockchain. This is a concept in which policies can be associated with money. These policies specify what happens after funds are transferred and can be checked, updated, or removed as the money is transferred. The success of the transfer is dependent on the availability of funds, an authorized spender, and if attached policies are fulfilled (e.g. the recipient is allowed to receive a fund for a particular purpose).

Pandey *et al.* (2019) developed two smart contracts on the Ethereum network that manage contributions to a crowdfunding campaign to prevent a range of issues caused by a major drawback, denying contributors control over the funds they raised. The money is controlled by the project manager, who can easily commit fraud.

The first smart contract is invoked when a new campaign is initiated. This contract generates an instance of the second smart contract, the campaign contract, which stores and manages the money raised for the campaign. After the amount of money is raised, the manager generates a payment request that must be approved by at least 50% of its contributors before the money is transferred.

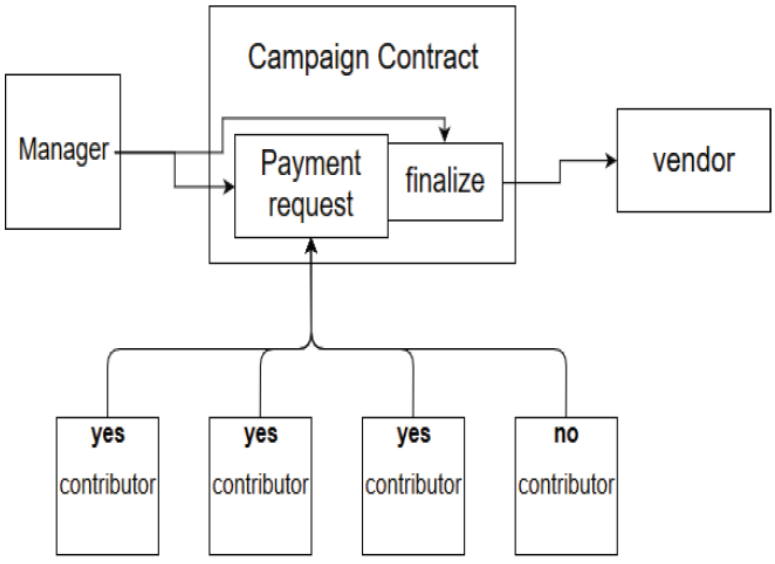

Figure 2.1: The core functionality of a campaign contract

(Adopted from Pandey *et al.*, 2019)

2.3 Automation

As they reflected on how automation works, Feltwell *et al.* (2019:9) advocated for automation in finance since it is envisaged to provide efficiency in supporting a range of benefits, such as account management and instantaneous calculations. They also note that automation can be a means of "enforcing intentions and providing consequence". This is related to setting up rules or 'recipes' using *Trigger-Action Pairs* to preserve the intention without relying on further human intervention. In action, this system helps change habits through the experience of consequences for certain behaviours.

The implementation of these *Trigger-Action Pairs* is known as IFTTT (if this then that), which is part of a wider growth in automation technologies. It specifically aims to remove the burdens of potentially "repetitive and time-consuming activities" through automation Feltwell *et al.* (2019:3).

2.3.1 IFTTT (If This Then That)

Feltwell *et al.* (2019) analyzed the novel integration between Monzo and 'If this then that (IFTTT)', where they reflected on automation to identify the potential of user-programmable money.

Monzo, a UK-based digital bank, has integrated some of its services with IFTTT. IFTTT is a web service that does not do anything independently but is the glue between two web services. Monzo's integration with IFTTT allows you to personalize how to manage money in a way that suits you. It works primarily with rules or a 'recipe' enforced through Triggers and Actions. The web service takes a specific trigger and a corresponding Action (Feltwell *et al.,* 2019). With this ability, you can create your own rules, called applets, to connect your Monzo account to different services you use. For example, you could have a rule to automatically 'reward' yourself every time you visit the gym by withdrawing a predefined amount of money from your 'gym visits' savings pot (Monzo, 2018).

Triggers
Any card purchase (above amount)
Any card purchase at specific merchant
Any new attachment added (e.g. image of receipt)
Any new blog post from Monzo
Actions
Move money into pot
Move money out of pot

Figure 2.2: Monzo + IFTTT Trigger actions

(Adopted from Feltwell 2019)

2.3.2 Smart Contracts

The concept of smart contracts is so complex that it transcends financial transactions. Omohundro (2014:19) identifies different capabilities of smart contracts ranging from catering for financial exchanges, insurance contracts, rental of digital storage, and computational power, which are easy to implement. If these contracts are automated, then decentralization becomes possible.

Monzo's integration with IFTT comes with limitations; its only product is currently a current account, and one can only move one's own money. For example, payment and bank transfers outside a user's account are not allowed. On the other hand, different projects demonstrate how immutable smart contracts applications, can grant devices financial autonomy on a larger scale. Unlike IFTTT, smart contracts have provided the basis for a new level of machine autonomy. According to Tallyn et al. (2018), smart contract technology represents an additional layer of machine autonomy, allowing machines to perform physical and cognitive tasks without human involvement. This gives them the ability to independently make executive decisions.

With the advantages of smart contracts, some authors still warn against the potential threats that smart contracts present. Pandey *et al.* (2019) warn that poorly written smart contracts can result in a loss of money. They also warn against non-deterministic contracts that will not produce the same output for a given input, hindering the network from reaching a consensus.

2.4 Systematic Literature Review

When conducting research, researchers need to outline existing literature before working on an area so that they avoid duplication of effort and are not biased in their work. This is the aim of a systematic literature review (Kofod-Petersen, 2014:1). This section presents a systematic literature review as a way to synthesize the relevant information available on the problem.

The study will find solutions from other studies, compare them with each other, and look at their implications on this research. To achieve this, multiple authors suggest similar methods; Brereton et al. (2007:572) provide a process that includes several distinct activities that can be grouped into three phases: *planning the review, conducting the review, and documenting the review*.

Okoli and Schabram (2010:6) present a guideline for conducting a systematic literature review with steps that are considered essential for research to be scientifically rigorous. This process can be seen in Figure 2.2 and will be followed by this study.

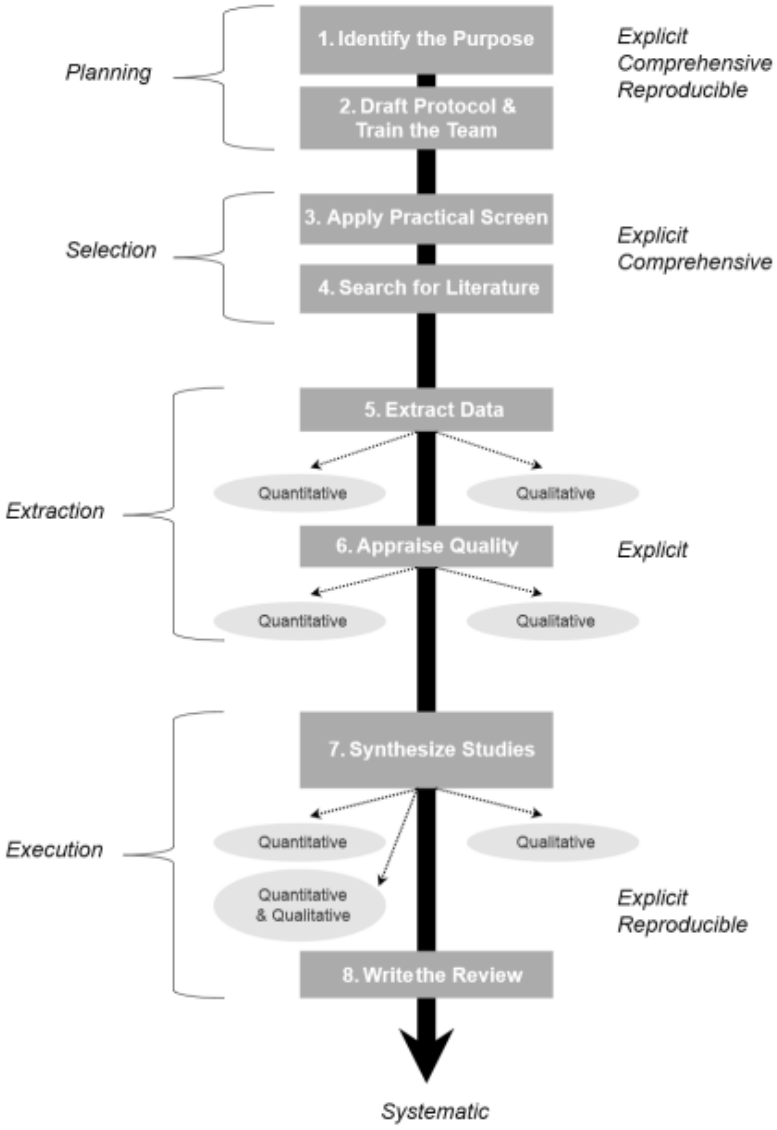

Figure 2.3: Systematic guide to literature review development

(Adopted from Okoli, 2015:885)

2.4.1 Systematic Literature Review Questions

Kofod-Petersen (2014:3) states in his paper that when conducting research In computer science, researchers typically want to know what solutions exist, how they compare, and the strength of the evidence and impact of those solutions. This systematic literature review aims to collect and investigate all possible currencies capable of programmatically enforcing contracts with limited human intervention. The systematic literature review research questions (SLRQ) in Table 2.1 were formulated to achieve this.

Table 2.1: Systematic Literature Review Questions

SLRQ1	What are the existing trusted currencies that are capable of enforcing contracts?
SLRQ2	How do the different solutions found by addressing SLRQ1 compare to each other with respect to methods/ and approaches?
SLRQ3	What is the strength of the evidence in support of the different solutions found in SLRQ1?
SLRQ4	What implications will these findings have on the creation of the proposed system?

2.4.2 Systematic Literature Review Protocol

The SLR protocol is performed during the planning phase (Figure 2.1). The aim of establishing a protocol is to minimize bias in the study by defining it in advance Brereton et al. (2007:572). Nightingale (2009:381) suggests that the first stage in conducting an SLR is to establish a protocol that defines the goal of the review, inclusion and exclusion criteria, the search criteria of the studies, and the plan of analysis. She also cautions against changing the established protocol since it could introduce bias.

2.4.2.1 Search Strategy

The search strategy was formed by gathering sources that may provide relevant studies for the review.

The study used the online research databases provided by the university (see Table 2.1). The study also includes relevant articles found from manual online searches where google scholar was the primary search engine.

Additional filtering excluded articles that had not been published and articles that had been published before 2014.

Table 2.2: Online search databases

Online Research Database	URL

Google Scholar	https://scholar.google.co.za/
ACM Digital	http://dl.acm.org
IEEE Explore	https://ieeexplore-ieee-org.ezproxy.cput.ac.za/
Emerald Insight	https://www-emerald-com.libproxy.cput.ac.za/insight/

The search process was restricted to only published journal articles and conference papers. The most commonly used search engine was Google scholar, and the most commonly used databases were ACM digital, IEEE Explore, and Emerald Insight (see table 2.1).

Other databases, such as ScienceDirect, were also included in the initial search but were later excluded for the following reasons:

- Non-unique search results – the result set was limited to studies that were already found in the other databases.
- The articles were not free.

2.4.2.2 Search terms

Once the search databases are defined, the next step is to define the search string with keywords that are relevant to SLRQ1, defined in section 2.4.1.

The search strings used to search for relevant studies are constructed as the following combination:

- Search string 1 (SS1): [*T2, C1*]
- **Search string 2 (SS2)**: ([*T1, C2*] and [*T1, C1*])

The search strings were constructed from Table 2.3.

The rationale behind the SS2 is that smart contracts are closely aligned. Autonomous enforcement of rules and financial transactions limits the application of smart contracts

to money which is relevant to the aim of this study. SS1 is also relevant to the study and is also the artefact produced by this study.

Table 2.3: Search terms

Terms	Categories	
	Category 1	Category 2
Term1	Financial Transactions	Smart Contracts
Term 2	Programmable Money	

2.4.2.3 Search results

Table 2.4 lists the results of studies obtained from the databases listed in section 2.4.2.1 using the search terms listed in section 2.4.2.2.

Table 2.4: Search results

Year	Online research databases					Total
	ACM Digital	IEEE	Emerald Insight	SpringerLink	Scholar	
2014	1					
2015						
2016	1					
2017			1	1		
2018	3	1				
2019	5	5	3		1	
2020	1	2	1	3	1	
2021		1	1		1	
Total	11	9	6	4	3	33

2.4.2.4 Study Selection

The study selected research papers where the articles were about programmable currency but were not necessarily limited to any trustworthy currency. These articles were filtered based on the relevance of the title, the abstracts, and the conclusions of these papers.

Using the inclusion criteria in Table 2.5, the selection process followed a three-stage process to filter the available articles:

- Abstract inclusion/exclusion screening
- Full-text inclusion/exclusion screening
- Full-text quality screening

Table 2.5: Inclusion criteria

Criterion Identifier	Criterion
IC1	Articles containing search terms defined in section 2.4.2.2 in the title of the paper.
IC2	Primary Studies
IC3	Papers that represent empirical results
IC4	Concerning programmable money
IC5	Concerning smart contracts, blockchain or cryptocurrencies in the context of enforcing terms and conditions.
IC6	Concerning the enforcement of a financial contract.

Table 2.6: Exclusion Criteria

Criterion Identifier	Criterion
EC1	Articles created or published before 2014
EC2	Articles that are not published
EC3	Articles that are not free to obtain through the CPUT library due to insufficient funding
EC4	Articles not written in English
EC5	Articles without experiments or empirical data for any form of programmable currency.
EC6	Articles that are not relevant to SLRQ1.

The study found several articles against the search strings defined in section 2.4.2.2. These findings were filtered based on the abstract screening as stage one of the 3 stage process mentioned above.

2.4.2.5 Quality Assessment

To address SLRQ3, the study focuses on the strength of the evidence presented by the studies in the review through quality evaluations. The purpose of this step is to exclude research that is not thematically related to the chosen field (Kofod-Petersen,

2012:5). Each study that has gone through the study selection process (Section 2.4.2.4) will be evaluated against the quality assessment criteria (QAC) in Table 2.7.

When performing the quality assessment using the QAC, there are three possible options for each criterion. If a criterion is satisfied, the response to that criterion is 'Yes', and if not, the response is 'No'. If the criterion is satisfied to some extent but not completely, the response is 'partly'. A score was assigned to each response ranging between 0 to 1. The scoring was as follows: *Yes = 1, Partly = 0.5, and No = 0*. The sum of the responses to each QAC was then calculated as a total score for the studies under review. The highest possible score for a paper is a 5. The minimum criteria for accepting a study is a total score of at least 50% of the highest possible score. With that being said, if the total score is less than 2.5, the paper will be rejected and not used in the systematic literature review.

Table 2.7: Quality Assessment Criteria

ID	Quality Assessment Criterion
QAC1	Does the research clearly state an aim?
QAC2	Is the study put into the context of other related studies and research?
QAC3	Are the design decisions of the artefact justified?
QAC4	Does the study present reproducible test data?
QAC5	Does the study present a reproducible experimental method?

2.4.3 Search Results

The total number of initial studies returned from the search was 33, as seen in Table 2.4. Through the application of a study selection process articulated in section 2.4.2.4, the result set was filtered down to the 16 articles listed in Table 2.8.

Table 2.8: Refined results

Study	Reference	Year	Title
S1	Pratyush Agarwal, Shruti Jalan, Dr. Ahijit Mustafib	2018	Decentralized and financial approach to effective charity
S2	Hyochang Baek, Junhyoung Oh, Chang Yeon Kim, Kyungho Lee	2019	A model for detecting Cryptocurrency Transactions with Discernible Purpose
S3	Jian LI, Shichao ZHU, Wen ZHANG, Lean YU	2020	Blockchain-driven supply chain finance solution for small and medium enterprises

S4	Martin C. W. Walker	2020	Do we need programmable money?
S5	Chris Elsden, Ludwig Trotter, Mike Harding, Nigel Davies, Chris Speed, John Vines	2019	Programmable Donations: Exploring Escrow-Based Conditional Giving
S6	Chris Elsdenm, Tom Feltwell, Shawn Lawson, John Vines	2019	Recipes for Programmable Money
S7	Muskan Vinayak, Saulo dos Santos, Ruppa K. Thulasiram, Parimala Thulasiraman, Srimantoorao S. Appadoo	2019	Design and Implementation of Financial Smart Contract Services on Blockchain
S8	Shivansh Pandey, Shivam Goel, Subodh Bansla, Dhiraj Pandey	2019	Crowdfunding Fraud Prevention using Blockchain
S9	Wenjun Fan, Sang-Yoon Chang, Shawn Emery and Xiaobo Zhou	2020	Blockchain-Based Distributed Banking for Permissioned and Accountable Financial Transaction Processing
S10	Xinyu Lei, Tian Xie, Guan-Hua Tu, Alex X. Liu	2020	An Inter-Blockchain Escrow Approach for Fast Bitcoin Payment
S11	Irvin Steve Cardenas, Jong-Hoon Kim	2018	Robot-Human Agreements and Financial Transactions Enabled by a Blockchain and Smart Contracts
S12	Rodrigo Couto de Souza, Edimara Mezzomo Luciano, Guilherme Costa Wiedenhöft	2018	The uses of the Blockchain Smart Contracts to reduce the levels of corruption: Some preliminary thoughts
S13	Ingo Weber, Mark Staples	2021	Programmable Money: Next-Generation Conditional Payments using Blockchain – Keynote paper
S14	Chris Elsden, Inte Gloerich, Tom Feltwell, Chris Speed, Belen Barros Pena, John Vines, Bettina Nissen	2020	Designing Futures of Money and FinTech
S15	Yakko Majuri	2019	Overcoming economic stagnation in low-income communities with programmable money
S16	Shangping Wang, Xixi Tang, Yaling Zhand, JuanJuan Chen	2019	Auditable Protocols for Fair Payment and Physical Asset Delivery Based tn Smart Contracts

The remaining 16 articles were evaluated against the quality assessment criteria listed in Table 2.7. Only articles that exceeded a total score of 2.5 were accepted into the study. This meant that a total of 10 papers were selected.

Table 2.9: Quality Assessment of Studies

Study	QAC1	QAC2	QAC3	QAC4	QAC5	Total
S1	1	0	0	0	0	1
S2	1	1	1	1	1	5
S3	1	1	1	1	1	5
S4	0	0	0	0	0	0
S5	1	1	0	0	0	2
S6	1	1	1	1	1	5
S7	1	1	1	1	1	5
S8	1	1	1	1	1	5
S9	1	1	1	1	1	5
S10	1	0	0	0	0	1
S11	1	0.5	0	0.5	0.5	2.5
S12	1	0.5	0	0	0	1.5
S13	1	1	1	1	1	5
S14	1	1	0	0	0	2
S15	1	0.5	0	0	0	1.5
S16	1	1	1	1	1	5

2.4.4 Results of the review

As indicated in Table 2.9, nine articles were accepted for this study. Upon closer look, the final selection revealed that eight of the ten articles performed experiments.

RQ1: A range of solutions exist in literature capable of autonomously enforcing agreements. Although not presenting a material solution but rather exploring a

possibility, S11 explores the design of a robot with an interaction model that enables it to engage in human-like financial transactions. The robot can also enter into agreements with human counterparts making this one of the most unique solutions. S2 only presents an artefact that indirectly addresses RQ1 in a smaller, narrower aspect, with an artefact that detects cryptocurrency transactions with discernible purpose. S5 examines programmable donations to third-party escrows where donations are released based on specified rules and conditions. S6 presents a qualitative study of a mobile banking app with a web automation service IFTTT (if this, then that) which works with rules or a 'recipe' that is enforced by triggers paired with corresponding actions where rules are triggered. S7 focuses on the clearing and settlement of contracts by implementing financial smart contract services on Hyperledger. S8 addresses a crowdfunding issue where contributors have no control over the money they have contributed, which is considered a major drawback. The study proposes a method that uses smart contracts to manage activities performed in a crowdfunding campaign. S9 proposes a blockchain-based banking scheme that leverages the integrated properties of blockchains to record and track permanent transactions. The issue being addressed is the lack of transparency in monitoring and tracking transactions in distributed banks, which results in a lack of support for auditing. S16 proposes an "auditable fair payment and physical asset delivery protocol based on smart contracts".

RQ2: The final selection revealed that nine of the ten articles performed experiments. The SLR revealed the blockchain as the predominant platform (S2, S3, S5, S7, S9, S13, S16). Smart contracts were also a favourite for a majority of studies that used the blockchain platform (S2, S5, S7, S13, S16). Three studies in the list were found to use the Proof-of-Work consensus protocol. The common reason for the strong adoption of smart contracts was autonomous and decentralized decision-making (mostly in financial transactions), addressing the fraud issue and improving auditability on financial transactions (S8, S16). S2 and S6 were the exceptions to studies that used blockchain as a platform for their respective solutions. S2 is a web service on a mobile platform, while S11 was an interaction model for a robot.

Table 2.10: Comparison of the different solutions

Study	Constraint	Method	Approach
S2	Machine Learning	Unsupervised learning	Unsupervised Learning Expectation-Maximization Algorithm for Clustering Datasets; Anomaly detection using Random Forest (RF).
S3	RBFT Consensus	Blockchain-driven SCF platform	Smart contract is created and loaded onto the blockchain.
S6	Web Services	IFTTT	Automation through trigger-action pairs that form recipes.
S7	BFT Consensus	Smart Contracts	Smart financial contracts designed for collateral services on Hyperledger Fabric, a permissioned blockchain.
S8	PoW Consensus Algorithm	Smart Contracts	A smart contract is created and loaded onto the Ethereum blockchain.
S9	PoA: Majority Decision distributed consensus.	Smart Contracts	A blockchain-based decentralized banking (BDB) scheme that leverages the built-in properties of blockchain to record and track immutable transactions.
S11	PoW Consensus Algorithm	Smart Contracts	A smart contract is created and loaded onto the Ethereum blockchain.
S13	PoW Consensus Algorithm	Smart Contracts	A smart contract is created and loaded onto the Ethereum blockchain.
S16	PBFT Consensus	Smart Contracts	Three types of smart contracts are designed to achieve reliable and fair payments among merchants, consumers, and logistics companies.

RQ3: All selected studies, particularly those presenting blockchain-based solutions, adhere to QAC5 defined in Table 2.7. S5, S10, and S13 are closely related and relevant to this study. These studies present solutions for conditional/fair payments and programmable money. They also present positive feedback.

S13 conducted a ‘Smart Money’ project to investigate the concept and usability of blockchain-based programmable money using smart contract technology. The project was motivated by the National Disability Insurance Scheme (NDIS) of Australia. This project was carried out on the Ethereum blockchain. Scalability issues are raised in the paper due to the PoW consensus algorithm. Nine participants were selected as end users for analytical and technical evaluation. The majority of the nine end users were highly positive about the features. The users valued the ability to make direct payments with certainty that the policies were fulfilled. This justifies the feasibility of the artefact proposed herein.

Although S6 contributes to SLQ1, it is not a blockchain solution and hence does not inherit the benefits of the blockchain platform. It also uses web services and not smart contract technology. S8, S11, and S13 use smart contract technologies provided by the Ethereum blockchain. However, the smart contract platform was known to have vulnerabilities at the time of writing (Bartoletti et al., 2021).

RQ4: Of the ten selected studies, seven utilize blockchain as a platform, and five of those seven also make use of smart contracts. The general theme is to rely on the built-in properties of blockchains, including decentralization and transparency, to support auditability and reduce fraud. Smart contracts contribute additional capabilities, such as enabling autonomous programmability of transactions such as conditional payments and clearing contracts.

The outcome of experiments conducted in these studies yields positive results – which positively contribute to future applications on the blockchain platform. The interest in the programmable money concept is also evident in the literature findings. The topic of programmable money is currently relevant, seeing that interest is recent seeing that most of the publications are not older than 2 years.

The review did not find evidence of self-contained intelligent tokens capable of self-enforcing contracts without the need for smart contracts. There is little evidence of using the PoS consensus algorithm, which solves the scalability and computational power requirements seen in the PoW algorithm. There is also evidence of vulnerabilities in the smart contract technologies used in these studies. These drawbacks present an opportunity to address the first objective of this study (Section 1.4) by choosing a blockchain platform that exploits a scalable consensus algorithm, such as PoS, with a smart contract platform with few known vulnerabilities.

2.5 Chapter Summary

This chapter reports on a systematic literature review that was conducted. The review was based on published literature related to the aim of this research (Section 1.4). A search was carried out in four online scholarly search databases: Google Scholar, IEEE Explore, ACM Digital, and Emerald Insight. The search was focused on solutions that address the concept of programmable money and self-aware currencies that can be trusted. The search string used (Section 2.4.2.2) returned a total of 33 articles.

Based on the selection criteria defined in section 2.4.2.4, the remaining number of articles had been reduced to 16, which were subjected to quality checks, leaving 9 viable studies.

The review found a range of solutions that include automation web services (IFTTT) integrated with Monzo bank. The limitation of this solution is that one can only act on one's own money; the solution is not on the blockchain and does not inherit the platform's benefits. Decision-making cryptocurrencies we found on the blockchain platform give evidence of the possibility of programmable money.

The review, however, found little evidence of solutions addressing the concept of cryptocurrencies that can be trusted to enforce contracts autonomously. The review also found little evidence of solutions that exploit the benefits of the PoS consensus algorithm. The question of scalability therefore remains. A multitude of bugs and attacks on existing smart contract platforms have also been found in the literature (Bartoletti et al., 2021).

Thus, the focus of this research is to address these issues. The details are described in the aim and objectives of this research (Section 1.4). Algorand, an open-source blockchain, supports layer 1 smart contracts. It is an open-source blockchain and exploits the benefits of the PoS consensus algorithm (Chen and Micali, 2017). Its smart contract layer (ASC1) aims to address smart contracts vulnerabilities and uses a non-Turing complete programming model that natively supports atomic transactions and custom assets (Bartoletti et al., 2021).

Based on these benefits, this research favours Algorand as the appropriate blockchain platform. This addresses the first objective of this study (Section 1.4). The next chapter reports on the approach used to solve this problem.

CHAPTER THREE : RESEARCH METHODOLOGY

The methodology chosen for this research was introduced in section 1.6 of the first chapter. This chapter will present a comprehensive guide to the steps used to produce this thesis. The chapter is structured against the research pyramid defined by Jonker and Pennink (2010:23). The thesis will make decisions based on the four "action" levels provided by the research pyramid. Paradigms, methodologies, methods, and techniques.

3.1 Introduction

According to Hevner and Ram (2004), “the objective of research in information systems is to acquire knowledge and understanding that enable the development and implementation of technology-based solutions”.

As stipulated in section 1.4 of the first chapter, this research aims to develop programmable money as a trustworthy virtual currency. Simply put, this research will produce an artefact. The utility and quality of this designed artefact must be rigorously demonstrated via well-executed evaluation methods (Hevner & Ram, 2004).

Design science research caters for this requirement through seven guidelines designed to assist researchers and reviewers in understanding the requirements for effective design science research (Hevner et al., 2004:11).

3.2 Research Pyramid

A research pyramid is presented by Jonker and Pennink (2010:23) to provide direction on how to structure their approach to the research. The fundamental premise is to put the researcher in a position to manage his or her research process and be held accountable for the choices made.

The pyramid is broken down into levels consisting of the : *research paradigm, research methodology, research methods,* and *research techniques*. Each of these levels is interconnected, starting from the research paradigm at a higher level of abstraction to the research techniques at a more concrete level. At each level, the researcher makes choices that align properly with the research based on the research questions.

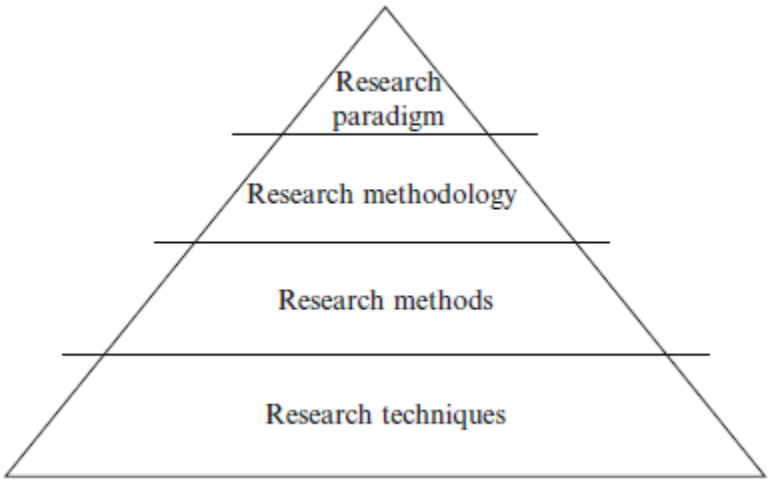

Figure 3.1 The Research Pyramid

(Adopted from Jonker & Pennink, 2009: 23)

3.2.1 Research Paradigm

The term paradigm has been used abundantly in research today. However, it is not an easy term to define while having been used in numerous ways. Bettis and Gregson (2001) define the term in its most generic sense, a basic set of beliefs that guides action in enquiry or research. In simpler terms, Jonker and Pennink (2009) define the paradigm as the way a researcher views 'reality' and is expressed in his basic approach.

There are currently several defined research paradigms, including positivism, constructivism, feminism (Bettis and Gregson, 2001), interpretivism, critical, emancipatory, pragmatism, postcultural, and postmodern. Most authors focus on positivism, constructivism, and pragmatism in today's paradigm discussions.

The most dominant of all paradigms are currently positivism. This is the oldest and most widely used approach. According to Lawrence (2014), positivism emphasizes the discovery of causality, careful empirical observation, and value-free research.

Constructivism is a paradigm that maintains that reality does not exist but is constructed by humans in relation to each other. The theory behind constructivism

suggests that humans construct knowledge from their experiences (Bada and Olusegun, 2015).

Pragmatism is action-based and is concerned with the interplay between knowledge and action. This usually applies to research types that intervene in the world and not merely observe. This would be appropriate if the said intervention is, for example, design research involving building artefacts (GoldKuhl, 2012).

This research is action-based and involves the creation of an artefact which is relevant to design research. Against the descriptions provided above, pragmatic research is a relevant paradigm to this research.

3.2.2 Research Methodology

Jonker and Pennink (2010:25) define research methodologies as "'a way' to conduct the research that is tailored to the research paradigm". It assumes a logical order the researcher must follow to achieve a result.

Hevner and Chatterjee (2010:5) define DSR as a research paradigm in which designers answer questions relevant to human problems by creating innovative artefacts. They also identify pragmatism as the root of design research.

In general terms, this research aims to create an artefact. The design science research methodology (DSRM) is based on the design and creation strategy (Figure 3.7) relevant to studies on the creation of an artefact. This research follows the DSRM and adopts the guidelines proposed by Hevner et al. (2004), which are discussed in detail later in this chapter.

Peffers et al. (2006: 93) introduce a design science research process (DSRP) model (see Figure 3.2), a conceptual process to be used as a 'mental model or template' and provide a nominal process to help researchers conduct design science research. Although structured in a sequential order, proceeding in a sequential order is not expected since the researcher can start at any step in the process and continue outward.

As highlighted in Figure 3.2, the *Problem Centred Approach* is the first entry point linked to *Problem Identification & Motivation*, the first step in the DSRP model. According to Peffers et al. (2006: 92), researchers proceed in this sequence if the

research idea originated from observing the problem or suggestions for future research in a prior study.

The study described in the thesis originates from observing a problem and analysing its importance; therefore, the *Problem Centred Approach* is used as the entry point.

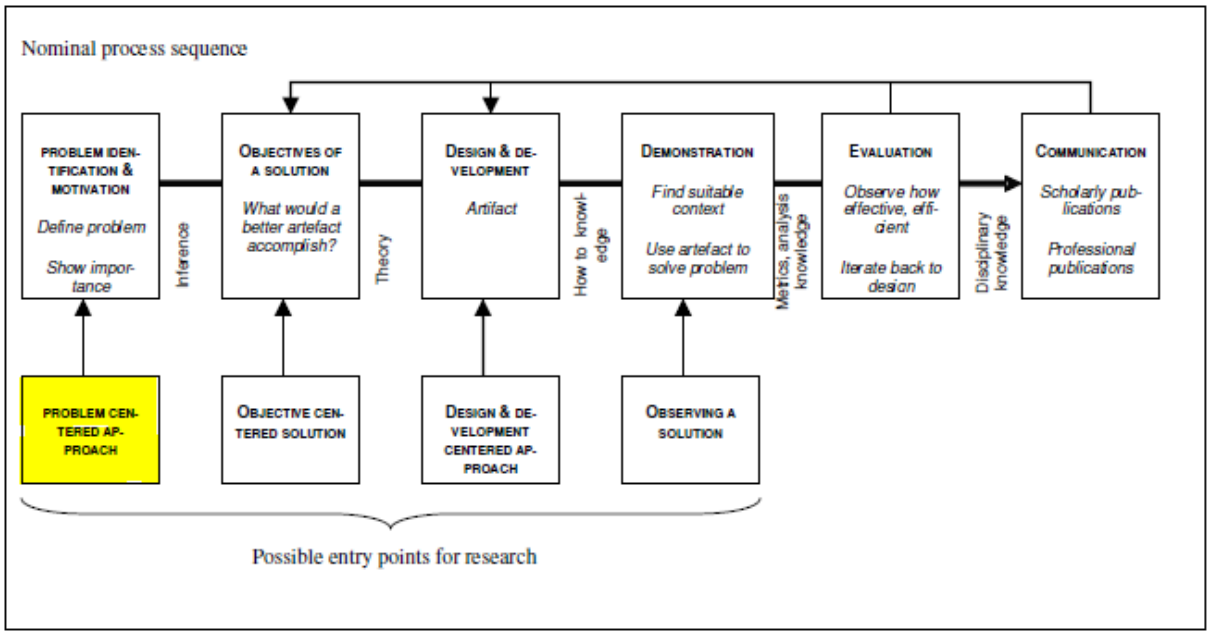

Figure 3.2: Design science research process (DSRP) model

(Adapted from Peffers et al.,2006: 93)

3.2.3 Research Methods

Research methods are more specific steps to execute a research methodology (Section 3.2.2). They can be defined as the tools and techniques used when conducting research (Walliman, 2017: 7).

This research will adopt the research methods suggested by Hevner et al. (2004) and Peffers et al. (2006: 89), who recommended the appropriate research methods for DSRM. These methods are listed in table 3.1:

Table 3.1: Mapping DSR activities to current research

Step	Activity	Description	Research context

1	Problem identification and motivation	The researcher defines the research problem and justifies the value of a solution (Peffers et al., 2006).	Chapter 1 sections 1.2 and 1.3 The research problem and background to the problem are articulated.
2	Objectives of a solution	The researcher derives the objectives of a solution from the identified problem (Peffers et al., 2006)	Chapter 1 section 1.4 Aims and objectives are articulated
3	Design and development	The researcher creates an artificial solution (Peffers et al., 2006)	Chapter 4 The overall design process, from requirements to testing and deployment, is documented
4	Demonstration	Demonstrate, through experiments, simulations, case studies, etc., how efficiently the artefacts solve defined problems. (Peffers et al., 2006).	Chapter 5 This chapter addresses how the artefact was subjected to different life-like simulations.
5	Evaluation	This activity involves comparing the observed results with the defined objectives to measure how well the artefact supports a solution to the defined problem (Peffers et al., 2006).	Chapters 5 & 6 Timeseries data collected from simulation results are observed and analysed to evaluate how efficiently the artefact supports the solution aligned with the research aim and objectives.
6	Communication	The results of design science research must be communicated efficiently (Peffers et al., 2006).	The research results will be communicated to the intended audience, in this case, within the thesis.

3.2.4 Research Techniques

According to Croker (2009:34), research techniques constitute a research method when they are commonly applied.

The main research technique applied in this research is observation. This is done mostly during the design cycle (Figure 3.6), where the evaluation of the artefact occurs under the design science research guidelines, which are explained in more detail later in this chapter.

The evaluation will be in the form of a simulation where quantitative data will be generated (see Figure 3.7). Quantitative data analysis methods were applied to the generated data, and mathematical and statistical analysis techniques were used. The quantitative data analysis and sampling techniques will be discussed in more detail in section 3.4.

3.3 Design Science Research (DSR)

What is design science research?

According to Hevner and Chatterjee (2010), DSR is a research paradigm that answers questions related to human problems through the design and creation of innovative artifacts. The aim is to learn about a design problem and its solution by creating an artefact.

Hevner et al. (2004) propose guidelines for conducting and evaluating DSR, which consist of seven elements summarised below in Table 1. These guidelines are based on the rationale that knowledge and understanding of design problems and their solutions can be gained through the design and creation of artifacts (Hevner et al., 2004; Hevner & Chatterjee, 2010).

Table 3.2: DSR Guidelines

(Adapted from Hevner et al., 2004; Hevner & Chatterjee, 2010)

	Criterion	Description
1	Design as an artefact	DSR requires the creation of an innovative, purposeful artefact
2	Problem relevance	For the specified problem, this artefact must be relevant and yield utility
3	Design Evaluation	A thorough evaluation of this artefact is essential.
4	Research contributions	The artefact must be innovative and therefore solve an unsolved problem or solve a known problem more effectively or efficiently.
5	Research rigour	The artefact must be rigorously defined, formally represented, coherent, and internally consistent.
6	Design as a research process	A search process must be included in the creation of the artefact, whereby a problem space is constructed, and a mechanism posed or enacted to find an effective solution
7	Communication of research	The results of design-science research must be communicated effectively.

3.3.1 Application of the guidelines to the proposed artefact

For this section, we will outline each of the guidelines tabulated in section 3.3.1 against the context of this research to articulate how this research adheres to the DSRM through the associated guidelines.

3.3.1.1 Design as an artefact

By pivoting the properties of a smart contract, this research addresses programmable money as a concept that can be applied in many different contexts. This research produces a trustworthy virtual currency as the artefact. This virtual currency autonomously enforces contracts during transactions.

3.3.1.2 Problem relevance

Literature Extract:

The requirement of a third party when entities engage in transactions is usually costly, time-consuming, and represents a single and central point of failure (Omran et al. 2018).

Gladden (2015) argues that through artificial intelligence, cryptocurrencies and machine ethics, it is possible to design intelligent cryptocurrencies that are capable of self-regulating their own use in a way that reflects human ethics or even human societies.

Through the popularity of blockchain technology and, most notably, smart contract technology, programmable money and conditional payments have recently become a topic of active interest in recent years (Elsden et al., 2019; Pandey et al.; 2019; Weber and Stables, 2021).

A literature review focusing on programmable money and the blockchain platform is conducted in chapter 2. Section 2.4 presents a systematic literature review where a comparative analysis is conducted on different solutions related to the problem defined in sections 1.2 and 1.3. The limitations and gaps in this comparative analysis contribute to the problem's relevance.

3.3.1.3 Design Evaluation

According to DSR, Artifacts must be rigorously evaluated in the laboratory and experimental environments before being released for field testing in line with relevant cycles (and its solution is acquired through the design and creation of an artefact (Hevner & Chatterjee, 2010).

During the design cycle, a balance should be maintained between the construction and evaluation efforts. These activities must be convincingly based on relevance and rigour.

The artefact can be evaluated at different levels of granularity, starting with unit tests at the lowest level. The evaluation can be scaled up to higher levels of granularity, such as lab tests in a controlled environment. There will then be scaled up to more robust generalization by increasing the population. An example of this is the application of PM in different social contexts such as finance, supply chain, government, healthcare, and education.

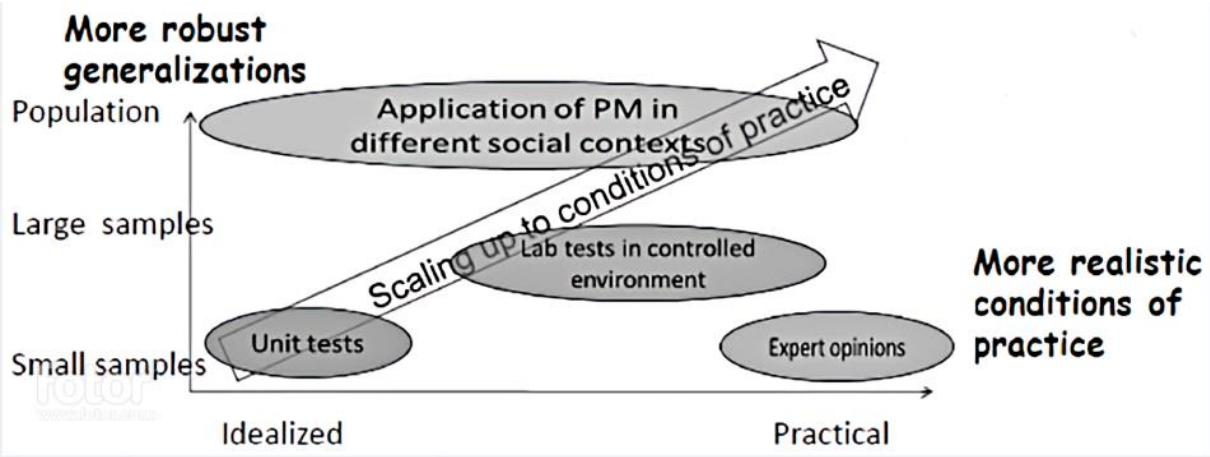

Figure 3.3: Scaling up the conditions of practice

(adapted from Wieringa, 2016)

The scaling to more realistic and practical conditions where the artefact is subjected to the experts in the field when it is deployed to the Algorand's BetaNet, an environment whereby different cryptocurrencies are deployed and used by blockchain, cryptocurrency, and smart contract experts.

Another example of scaling up to conditions of practice would be data gathered on the performance of the artefact when funds are allocated to a single student, followed by data allocated to an entire college for larger-scale evaluation (Figure 3.4).

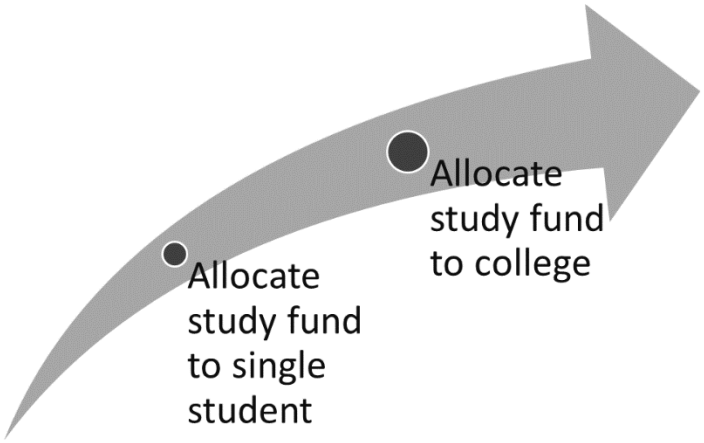

Figure 3.4: Example scenario of scaling up to more realistic conditions

The evaluation in this research is conducted through a simulation that consists of functional and nonfunctional tests. During the functional tests, the artefact is subjected to simulated transactions on programmable money crypto tokens to observe how efficiently the artefact enforces contracts associated with the tokens.

During the non-functional tests, the artefact is flooded with many simulated transactions to observe how the artefact performs under stress and better understand how the artefact would perform under realistic conditions. The simulation and findings are reported in Chapter 5 and Chapter 6.

3.3.1.4 Research Contributions

The contributions of the study are as follows.

- Software contributions
  - PM developed artefact.
  - Library for long-running smart contracts to Algorand BetaNet Blockchain.
- Empirical data about testing PM in a controlled environment.
- Data about challenges and best practices required to deploy PM.
- Data about the limitation of PM and requirements for large-scale deployment, including performance metrics of the various methods to be used and tested.

- Application of PM as a design science artefact to the problem context.
    - Information detailing how and why the DSR artefact succeeded/failed to solve the problem.

3.3.1.5 Research Rigor

- Software development process
    - SDLC: Agile development methods
    - Software engineering methodologies (Figure 3.5).
- Evaluation methods
    - Performance metrics
    - Socio-technical considerations
    - Requirements contract
    - Simulation requirements

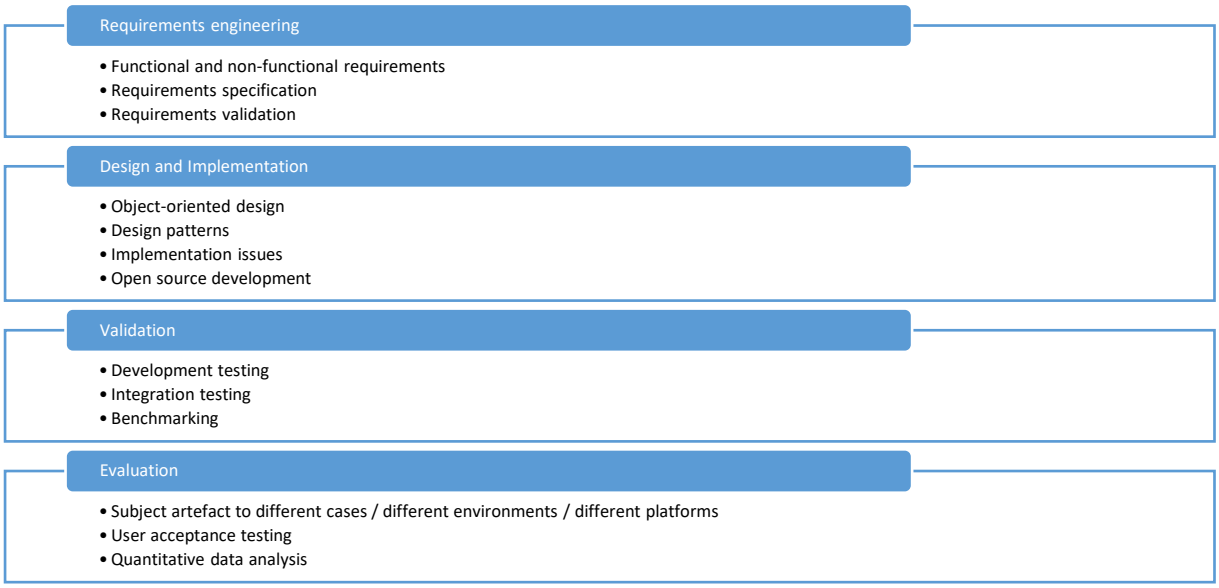

Figure 3.5: Software engineering methodologies

3.3.1.6 Design as a search process

Design is essentially a search process that seeks to effectively solve a given problem.

Construction of the artefact is not a standalone process. According to DSRM, there is a relevance and rigour cycle (Hevner & Chatterjee, 2010). From the relevance cycle,

we derive the problem and the requirements. The details can be found in guideline 2. From the knowledge base, we will get the best practices to maximize the precision of the artefact being developed.

Therefore, there will be a cycle in which development and evaluation are balanced. There will be a continuous alternation between the knowledge base and the development cycle to find expert information, such as development methodologies. There will also be a continuous alternation between the evaluation task and the environment to determine whether the artefact solves the problem found in the environment and whether the artefact is still relevant to the problem. If it does not, the development phase will be revisited to address any shortfalls.

3.3.1.7 Communication of research

- The initial proposal is reviewed by two reviewers and presented to the department during a proposal defence.
- The results of the research will be communicated to the intended audience, in this case, within the thesis and published articles.
- The open-source community will receive communication about the library's contribution.

3.4 Data Collection

In section 3.2.2, the problem-centred approach was defined as the appropriate entry point for this research. This is for scenarios in research where the research idea originates from observing the problem or suggestions for future research in prior studies (Peffers et al., 2006: 92).

The problem in this research was identified by observation and described in detail in sections 1.2 and 1.3. A systematic literature review is conducted in section 2.4, where prior studies addressing the problem were collected. A comparative analysis (section 2.4.4) was done to identify the quality, limitations, and gaps found in the current literature. The identified gaps and limitations contributed to justifying the aim of this study (Section 2.5).

3.5 Data Generation and Analysis

Under the guidelines of DSR (Section 3.3.1), the artefact underwent rigorous evaluation. During the evaluation, a simulation was conducted to subject the artefact to life-like scenarios. This is aligned with the evaluation in DSR guidelines. Peffers et al. (2006:92) state that DSR is evaluated by observing and measuring how well the artefact supports a solution by comparing the objectives of a solution to the actual observed results. The data generation method during the simulation was observation (see Figure 3.7).

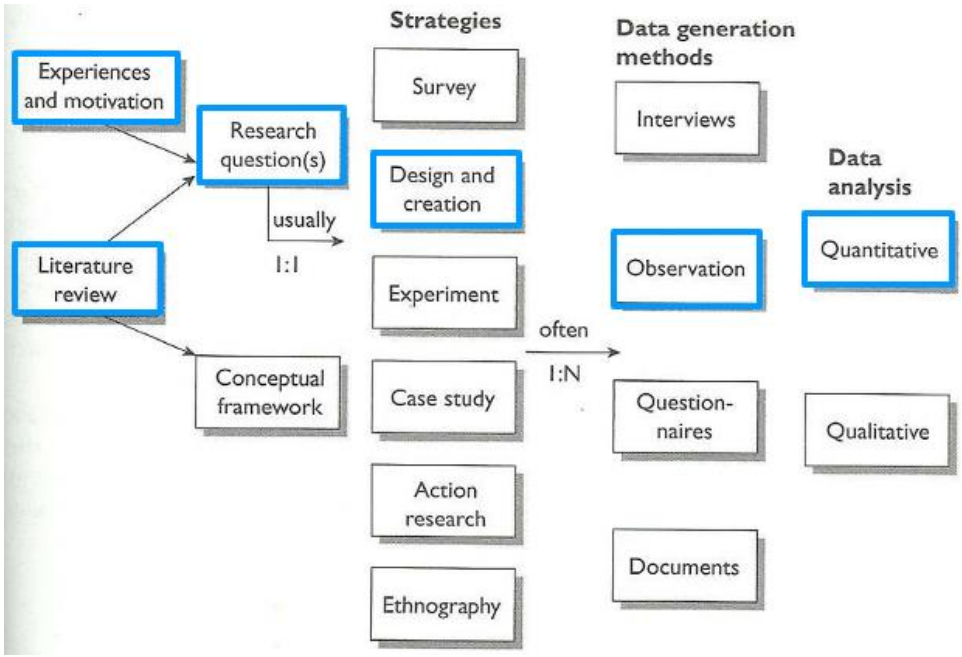

Figure 3.6: Research process followed

(Adapted from Oates, 2006:33)

As highlighted in the Oates model (see Figure 3.6), quantitative data analysis was conducted in this research. The quantitative analysis was conducted against appropriate metrics as part of the evaluation activity of the design cycle in DSR cycles. As illustrated in Figure 3.6, the analysis occurred in a cycle between the *Build Deign Artefact* activity and the *Evaluation* activity.

Sampling

According to Yilmaz (2013), quantitative methods generally require large representative samples to allow the researcher to generalize from their sample. In support of this, Fowler and Lapp (2019:61) state that even though large samples supply no guarantee of accuracy, larger sample sizes are typically more representative of the population you are studying.

For this research, the unit of analysis will be the data generated by the simulation, where large amounts of data will be generated to best represent the performance of the artefact. During the evaluation phase, the artefact was subjected to functional and non-functional tests and the observed performance was generated as the data used for analysis. The Siege http load testing tool was used as part of the non-functional tests to benchmark the artefact and generate the performance data used for analysis.

3.6 Summary

This chapter mapped a detailed research process using the research pyramid that Jonker and Pennink (2010) presented.

The chosen methodology was the design science research methodology. It is based on the pragmatic paradigm. The research outlines the seven guidelines for conducting design science research outlined by Hevner et al. (2004) and how they apply to this research. Quantitative data was collected and analysed. This data was collected in a simulation conducted during the evaluation of the artefact.

CHAPTER FOUR : DESIGN OF PROGRAMMABLE MONEY ON ALGORAND BLOCKCHAIN

This chapter fully documents the overall design process, from requirements to testing and deployment. It includes all the design decisions made and the trade-offs. The design decisions include the design goals (Section 4.2) and the design requirements (Section 4.3). Section 4.4 presents a system overview where the components of the system are outlined. Details about the deployment of the system are detailed in section 4.5. The resources required are outlined in section 4.6.

4.1 Introduction

The rationale behind the development of programmable money was influenced by several factors mentioned in Chapter 1 and Chapter 2. Through the systematic literature review, the background knowledge of the problem and what has been done was explored. This feeds into the problem relevance, the second guideline of the DSR guidelines presented in Chapter 3.

This chapter builds upon the findings of the systematic literature review and the research methodology by documenting the design process from requirements to testing and deployment. This includes design decisions and trade-offs.

4.2 Design Goals

This system design aimed to design the proposed artefact to fulfil the objectives articulated in section 1.4. The proposed artefact is programmable money as a new virtual currency that pivots the capabilities of smart contracts to enforce agreements autonomously. In simpler terms, the proposed solution should be an environment of conditional payments where funds transfer is only successful when predefined agreements are met. The proposed solution includes a cryptocurrency token with an attached agreement. This agreement is determined by the current owner, who donates the token to a spender. The spender may spend the token by making a payment request that invokes the smart contract deployed on the blockchain. The smart contract ensures that the payment request is not violating the attached agreement associated with the token. The smart contract updates the token ownership and

attached agreement if the agreement is not violated. The transaction is then written to the blockchain.

4.3 Design Objectives/Requirements

The reason behind the requirements gathering is to assist in achieving the purpose of the research (Section 1.4). After conducting a background and systematic literature review, the design requirements were established and formalized as DR1 to DR6.

The design objectives are motivated by technological innovation and represent high-level requirements. Given that the proposed artefact pivots the capabilities of a smart contract, the design requirements below are centred around a smart contract. Built-in properties, such as autonomous execution, are assumed to be included in the solution.

- DR1 - Efficient and secure payment platform.
- **DR2 – Embedded agreement:** The contract *must* be embedded in the money.
- **DR3 – Embedded behaviour:** The system *must* efficiently enforce a predefined contract.
- DR4 - Better visibility and transparency of embedded agreements.
- DR5 - Configurability of embedded agreements: The embedded contracts must be reconfigurable
- **DR6 – Generic context:** The system *must* be generic and usable in different contexts.

4.4 System Overview

This section describes a detailed design of the artefact and explains the design choices that led to the proposed solution.

The discussion of programmable money has been a topic of active interest in recent years (Agarwal, Jalan and Mustafi, 2018; Elsden *et al.*, 2019; Wang *et al.*, 2019; Fan *et al.*, 2020; Cunha, Melo and Sebastião, 2021; Weber and Staples, 2021). It is powered by smart contracts and is about embedding behaviour in the money. For example, the execution of payments can be executed when certain criteria are met (Cunha et al., 2021).

By pivoting the properties of a smart contract, this research addresses programmable money as a concept that can be applied in many different contexts. As an aid to the research, programmable money as a trustworthy virtual currency is the proposed artefact. This artefact deals with the issue of trust in financial transactions by autonomously enforcing agreed terms and conditions with each transaction performed.

4.4.1 Conceptual Model

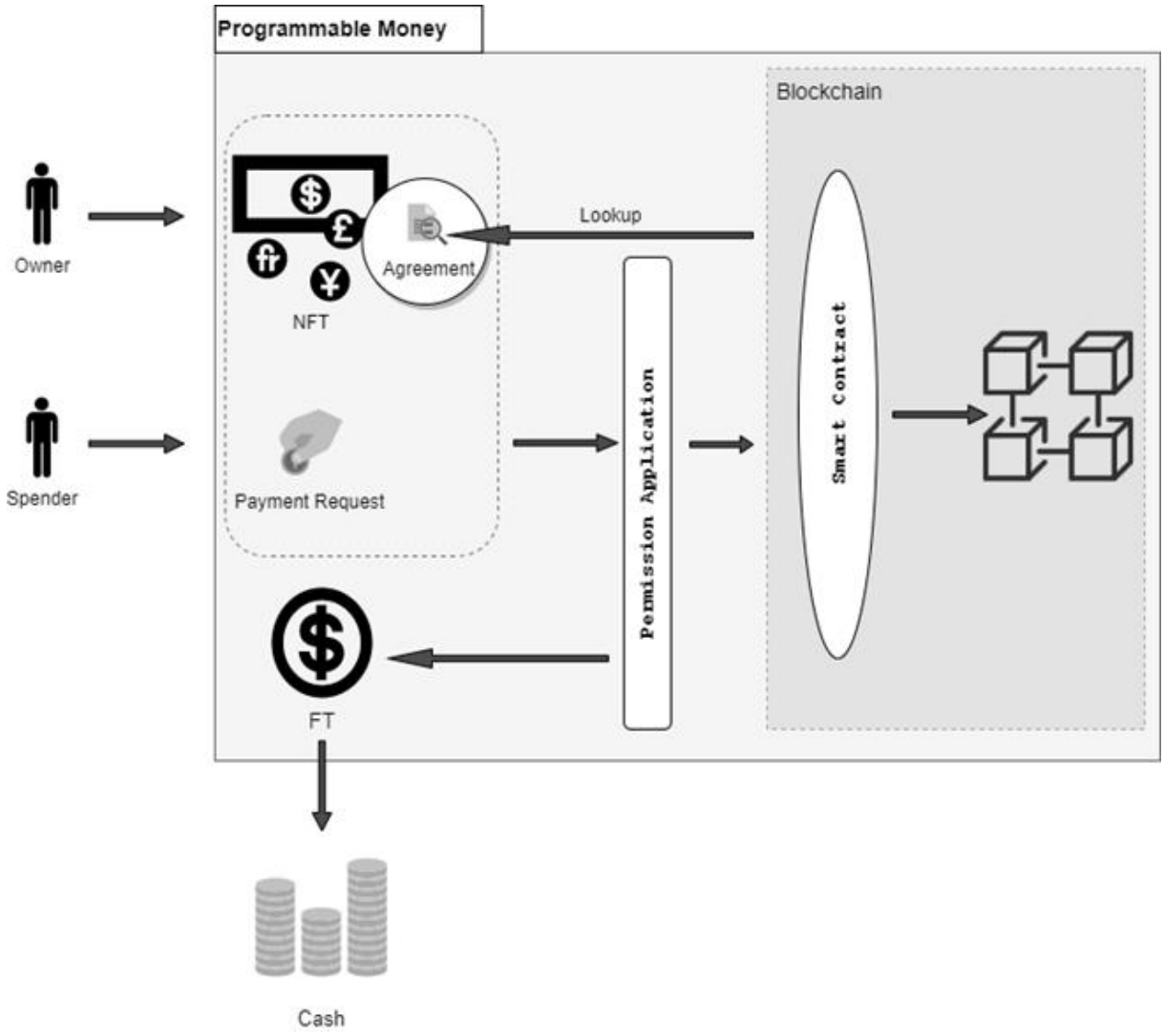

Figure 4.1: Abstract Representation of the Programmable Money solution

4.4.2 System Components

4.4.2.1 Tokens with dynamic conditions

The system must include cryptocurrency tokens (Section 4.4.2.3) with attached conditions governing the spending requirements.

4.4.2.2 Smart Contract

The system must include a smart contract application that governs the flow of PM crypto tokens.

The system intends to simply guarantee that funds are spent only for their intended purpose. Smart contracts are the cornerstone of this by providing logic that gives birth to programmable money. On a high level, a smart contract should ensure the following:

- a transfer only succeeds if the funds are available.
- the spender is authorized.
- agreements that govern transactions involving the token are not violated.

The Algorand architecture comprises two types of contracts, stateful and stateless. All transactions in Algorand must be signed, and stateless smart contracts fulfil the function of a signature authority for transactions. Although stateless contracts serve a purpose in the artefact, stateful smart contracts are the programs of interest (Smart Contract Overview - Algorand Developer Portal, 2021).

Stateful contracts live on the chain and are triggered by special Algorand transactions. These contracts have specific global values and per-user values, and they control the logic of how these values are modified. This provides longevity, a property that normal money - digital or otherwise - does not currently have. These stateful contracts can be combined with other Algorand features to produce even more functionalities in applications. One of these Algorand features includes atomic transfers (Smart Contract Overview - Algorand Developer Portal, 2021).

4.4.2.3 Atomic Transfer

Atomic Transfers are the logical grouping of transactions that form part of a larger transfer. Each of the transactions in this grouping is assigned the same group ID. All

of the transfers in an atomic transfer have to succeed, or the transfer fails. This ensures that both parties in a transaction receive what they agreed to and removes the requirement of a third party (Smart Contract Overview - Algorand Developer Portal, 2021).

Atomic transfers can facilitate the fulfilment or execution of agreements, for example, group payments, where everyone pays or no one pays. The global state of this group payment will continue to be managed by the logic of a smart contract.

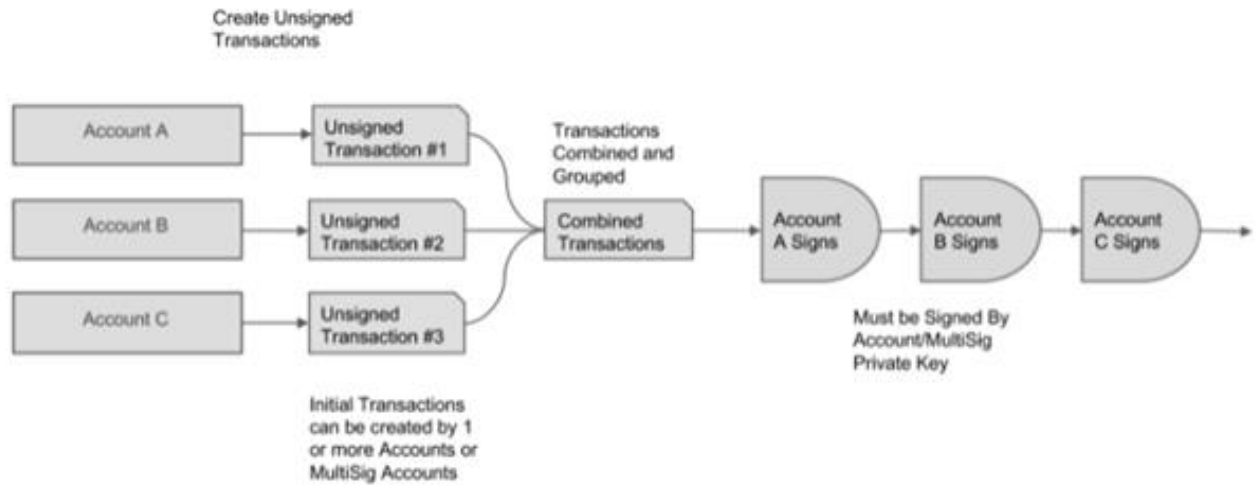

Figure 4.2: Automatic Transfer Flow

(Adopted from Smart Contract Overview - Algorand Developer Portal, 2021)

4.4.2.4 Assets

The assets traded in the abovementioned transactions and transfers are officially named Algorand Standard Assets (ASA) on Algorand. These ASAs can represent stablecoins, loyalty points, and system credits. ASAs are not restricted to monetary value but can represent unique assets like a deed for a house or unique parts on a supply chain (Smart Contract Overview - Algorand Developer Portal, 2021). For this research, ASAs and tokens are used interchangeably. Nonfungible tokens called Stable Rands (STR) are created as part of the solution.

4.4.2.5 Token Owner

The system must include a token owner who owns tokens, determines the content of their attached conditions, and can donate tokens to a spender (see DR4). The token owner may also have the role of *Spender*.

4.4.2.5 Spender

The system must include a should cater for a spender who attempts to spend the programmable money tokens by generating a payment request. The spender is any entity in the system that is allocated money by the token owner. These tokens are allocated with an attached contract. The spender is a generic entity that can exist in any social context. A few examples range from a donor (consider a recipient of a charity fund as an example), a student who earns a financial aid grant, and a financial officer who oversees all financial transactions in a business.

4.5 Deployment

4.5.1 Virtual Machine

Algorand Virtual Machine

Programmable money, which pivots smart contracts, will be compiled on the Algorand Virtual Machine (AVM). The AVM supports smart contracts with Turing-complete languages, and additional features. It runs on every node in the Algorand blockchain and contains a stack engine that executes smart contracts and smart signatures, programs that are used to sign transactions (Algorand, 2021).

4.5.2 Server

Algorand’s BetaNet is an app-rich network that will contain new features that developers are currently working on. BetaNet came into existence because the vision for TestNet was for the protocol to always mirror the MainNet. The TestNet is therefore reserved for applications with realistic network conditions before being deployed to the MainNet. As these apps mature, they are slated to eventually be deployed to the

TestNet and, eventually, the MainNet environment – the production network. The development of the artefact in this study serves as a proof of concept (POC), and it will be deployed on a sandbox environment pointing to BetaNet.

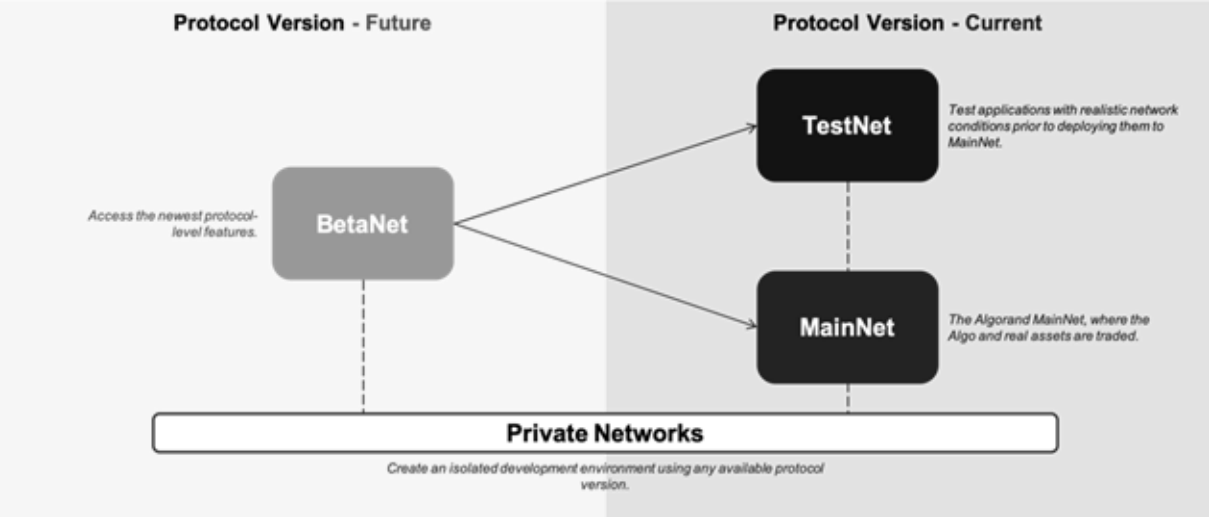

Figure 4.3: BetaNet, TestNet, and MainNet

(Adopted from Fustino, 2020)

4.5.3 Infrastructure

The developed artefact was evaluated through a simulation on a docker sandbox installed in an Ubuntu Virtual machine (VM). The set-up scripts and simulation scripts were created on the Ubuntu VM. In the sandbox, the Algorand blockchain and smart contract application were deployed.

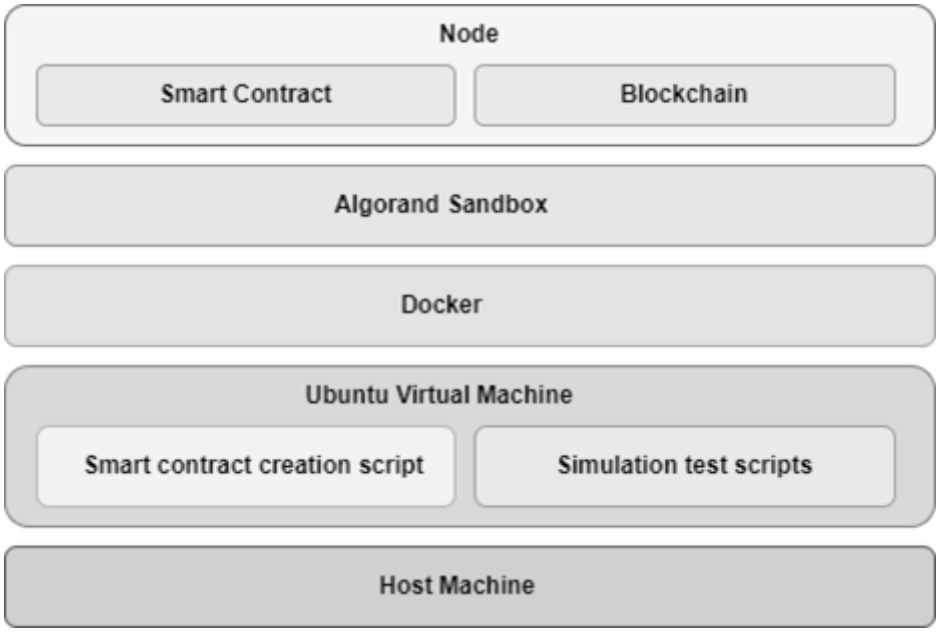

Figure 4.4: Infrastructure where the artefact is deployed

4.6 Resource requirements

Table 4.1: Resource requirements

Tool	Description
Ubuntu Virtual Machine	OS platform for development environment
Java > 11 SDK	Java 11 or higher: development kit
IntelliJ	IDE to be used
Algorand BetaNet	Blockchain network reserved for applications in the development phase
Teal	Language used for Algorand smart contracts.
Purestake Algorand Rest API	Provide access to the Algorand network.
Algorand Sandbox	Provides a private network for local testing and a quick way to create an Algorand development environment.

Siege	An open-source regression test and benchmark utility.

4.7 Chapter Summary

Chapter four expanded on the methodology discussed in chapter three. This chapter discussed the design goals and objectives. The overview of the system was also discussed, including the conceptual model, and system components were discussed in full detail. The deployment and server details and general resource requirements were also discussed.

CHAPTER FIVE : EVALUATION OF PROGRAMMABLE MONEY THROUGH SIMULATION

5.1 Introduction

The previous chapter discussed the concept of programmable money as an artefact and the design and breakdown of each component involved. Chapter 5 discusses how the experiment and evaluation of the artefact were carried out in detail. A simulation consisting of functional and non-functional tests is undertaken. Inputs (such as test parameters and metrics) and the observed results are reported in this chapter.

5.2 Problem Domain

Hevner & Chatterjee (2010) define DSR as a research paradigm in which human problems are answered via the creation of innovative artefacts. Programmable money as a concept aims at autonomously enforcing contracts associated with money so that the fund's purpose is not violated.

The relevant problem domain, in this case, is deduction. The theme idea for the simulation is whitelisting, a concept similar to the disbursement of funds in the NSFAS system. In this system, the NSFAS is the token owner, who will allocate funds to the student by dispersing tokens that have attached agreements. The attached agreements are enforced autonomously with each transaction. The data was collected through observations, and deductions were formed through the analysis of the collected data.

5.3 Simulation Outline

This simulation will run on a Linux virtual machine running Ubuntu 20.4 as an OS.

The technologies to be used are listed in section 4.6. Additionally, bash scripting (packaged in Ubuntu OS), a type of shell scripting, is primarily a tool to interact with the Algorand platform through the sandbox. Shell scripts are responsible for creating tokens, and smart contracts (also called applications) that manage the tokens, opting-in accounts to the smart contracts and assets, and creating and interacting with these smart contracts and assets by attempting to send transactions to the network.

5.3.1 Participants

There are no human participants in the simulation. The participants are the Algorand accounts created in the sandbox private network. These accounts have opted-in / registered to the created tokens and smart contract applications. The shell scripts (Appendix E-H) will attempt to send transactions to the blockchain on the private network on behalf of these accounts. These transactions simulate normal human transactions if an asset is purchased. This purchase has an asset transfer transaction and a corresponding payment transaction for that asset. The smart contract application is invoked to facilitate this transaction and ensure that all the rules and associate policies are adhered to.

5.3.2 Experiment Set up

5.3.2.1 Stateful smart contract application

The main smart contract application manages the events where ASA's (Section 4.4.2.4) as the PM crypto token is involved in a transfer transaction. Permissioned tokens are specially designed tokens with built-in control mechanisms that limit the token's usage based on additional requirements defined by the business strategy and regulations (Zaremba, 2021). These tokens support a range of use cases, such as implementing transfer restrictions, managing the freezing of assets to ensure no transfers are issued before they are allowed, and creating whitelisted accounts approved to buy and hold accounts.

The stateless smart contract in this study demonstrates the following:

- Manage the freezing and unfreezing of the token mentioned above
- Restrict asset transfers to ensure adherence to pre-determined conditions (e.g., atomic transfers (Section 4.4.2.3)
- Create whitelist accounts that can participate in the trade of the asset mentioned above.

Smart contract creation

The name of the smart contract is the permission app. The first part of the app is the creation which stores global variables and performs validations that confirm that conditions associated with application creation are met. These conditions ensure that:

- The correct number of arguments are passed in the application call
- The global variables are stored for further use (e.g., the identity of the permissioned token to be managed by the current application)
- Determine which section of the app to branch to if the application call is not a creation

Whitelisting of accounts

This part of the application is responsible for adding a whitelist attribute to certain accounts. Whitelisted accounts are the only accounts with the ability to transfer the permissioned tokens (provided other conditions are also met).

The smart contract validates that the correct number of arguments are passed in the application call and that this transaction is only performed by the app creator, who is also the escrow account (Section 5.3.2.2).

If successful, the target account is assigned a whitelist attribute, a required attribute if the account is involved in a transfer of the permissioned token. The global whitelist count is incremented.

Transfer

The transfer part of the application handles the movement of the permissioned token between whitelisted accounts. The transfer is atomic (Section 4.4.2.4) and includes three transactions a) an application call; b) a transfer of the permissioned token from whitelisted account A to whitelisted account B; c) a transaction fee payment. The standard rules of an atomic transfer apply. Either all or none of the transactions in an atomic transfer are successful. For a transfer transaction, the smart contract makes the following validations:

- The group size in the atomic transaction is 3 (see above)
- The order of transactions is a) application call, b) transfer, and c) payment
- The remaining funds in the Sender’s account are not maliciously moved to a different account

- The authorization (which authorized all future transactions) is not being changed
- The correct token is being transferred
- The max limit of permissioned tokens that can be transferred is not breached
- The Sender and the Receiver account are whitelisted
- The transaction fee is paid

5.3.2.2 Escrow Account

The clawback escrow account is a stateless smart contract (see Figure 5.1). It acts as an autonomous signatory responsible for signing transfer transactions and making initial validations. These validations include ensuring that this is a token transfer and that the permission smart contract application is being called (Section 5.3.2.1). The transaction will fail in the initial phase if any of these validations fail. Otherwise, the permission app is invoked for further processing.

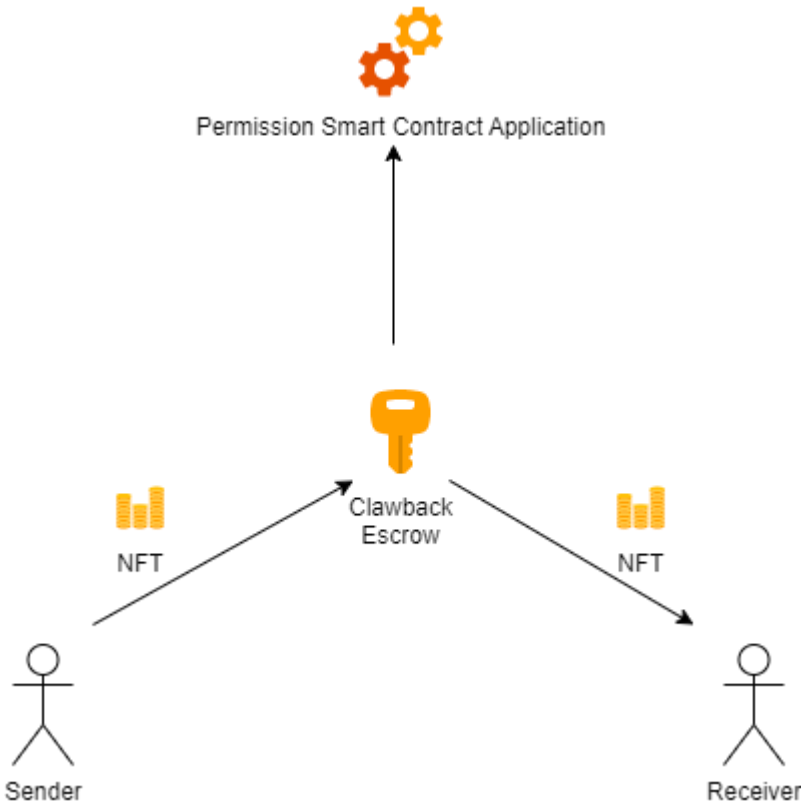

Figure 5.1: Transfer Transaction

5.3.2.3 Creation Script

The creation script (see Appendix A) is responsible for setting up the sandbox private network for testing. It reads one participating account from each node. It creates the asset (Section 4.4.2.4) to be traded. It installs the smart contract application described in section 5.3.2.1 to the blockchain. The created asset and application are then linked to the participating account.

5.4 Test cases

Wieringa (2016) states that the implementation of artefacts from idea to practice starts from small laboratory conditions, i.e., start development and test on the context of a specific group to ensure lab credibility and move to a larger group sample to maximize the street credibility. The main goal of this research was to verify the idea through simulation.

Below are the test case scenarios that were executed during the experiment:

- Transfer that conforms with contractual agreements
- Transfer without transaction fee payment
- Transfer to a non-whitelisted account
- Transfer from a non-whitelisted account
- Transfer to a non-whitelisted account registered to trade to STR in the programmable money system
- Transfer from a non-whitelisted account registered to trade to STR in the programmable money system
- Transfer that violates the maximum allowed amount to transfer STRC units
    - For the purpose of this test, we have set the maximum transfer limit to 50 units
    - A random number generator was used to determine the unit amount with each transfer
    - The random number was set to range between -50 and 100.
- Performance simulations
    - Increase in the amount of requests
    - Increase in the amount of concurrent users

- Invoke multiple services simultaneously.

5.5 Functional Simulation Results

A total of 12 accounts were used for this during the simulation.

Table 5.1: Algorand accounts used in simulation tests

Account	Address
Account A	NCKAXR3NGA245N3R5HP3CRKOQ6PIQYVSLJHSIHO5DR6D2NONXVFJQOM3PA
Account B	GFPY3O3ZH5TJ7QT7RRCDN2C65DSPALNFXF3RCDYLBVIX2GFMWHVZF67OZQ
Account C	HKW4NQX7H26WSOVNV7HAYPZYB6SSJ5HFZ6KL4GAOYEOGRXOBEUPOB324OE
Account D	BYYNYAD35RA6OIFCWBNEB2DTRXJHTM54JVIKJFZRW2XS6HGKTA4TNY5AJA
Account E	Y5HLLPYPQUWIIQRSTNQQ7FRQWH2UFFJMSWFGZG2N6VNYLMUHIASRGLKXP4
Account F	2AC7ZVQ2UD5MNFQFLERDCC4FJAOM5MQQRT5AS4XTT5DE4UISX54XIFUBGA
Account G	Q3A4RGB2BJ2X5PBIS6HL6DZ3GXXCEV6ZB7BE2OWDGL62C3GQ6CONTLSTNQ
Account H	XGZTTWRJMR6EJZHUY3PWO6N3ZM367IA73YK4JZHCD4OZMXCS5UIRZT2QXU
Account I	G3OJUQDFRSAUSIVSMVNAXRRTSUIFNQWSKHS2WUJJTGOYRQIWQLGEFUESVQ
Account J	NCKAXR3NGA245N3R5HP3CRKOQ6PIQYVSLJHSIHO5DR6D2NONXVFJQOM3PA
Account K	APZSGXKEWISGBSVQR2NK4JS4ILMKJW2FGQQ5E5LAJK3XOPA2SGE3ANY57M
Account L	IRFFUEKP5AF5H4VZTBQJYXU6KPOX6GQP4Y5D6MHNBAVO2GZ2S3VGAOBHGA

Below is the associated metadata for the accounts above (see Table 5.1)

Table 5.1: Account metadata

Account	Opt-in	Whitelisted	Escrow
Account A	Yes	Yes	No
Account B	Yes	Yes	No
Account C	No	No	No
Account D	No	No	No
Account E	Yes	No	No
Account F	Yes	No	No
Account G	Yes	No	No
Account H	Yes	No	No
Account I	Yes	Yes	No
Account J	Yes	No	No
Account K	Yes	No	No
Account L	N/A	No	Yes

5.5.1 Payment to non-whitelisted account

Payment between non-whitelisted accounts is a transaction that attempts to transfer STR from account A to Account B where one of the accounts is a) not whitelisted, b) not opted into the smart contract, and c) is not associated with the asset being transferred.

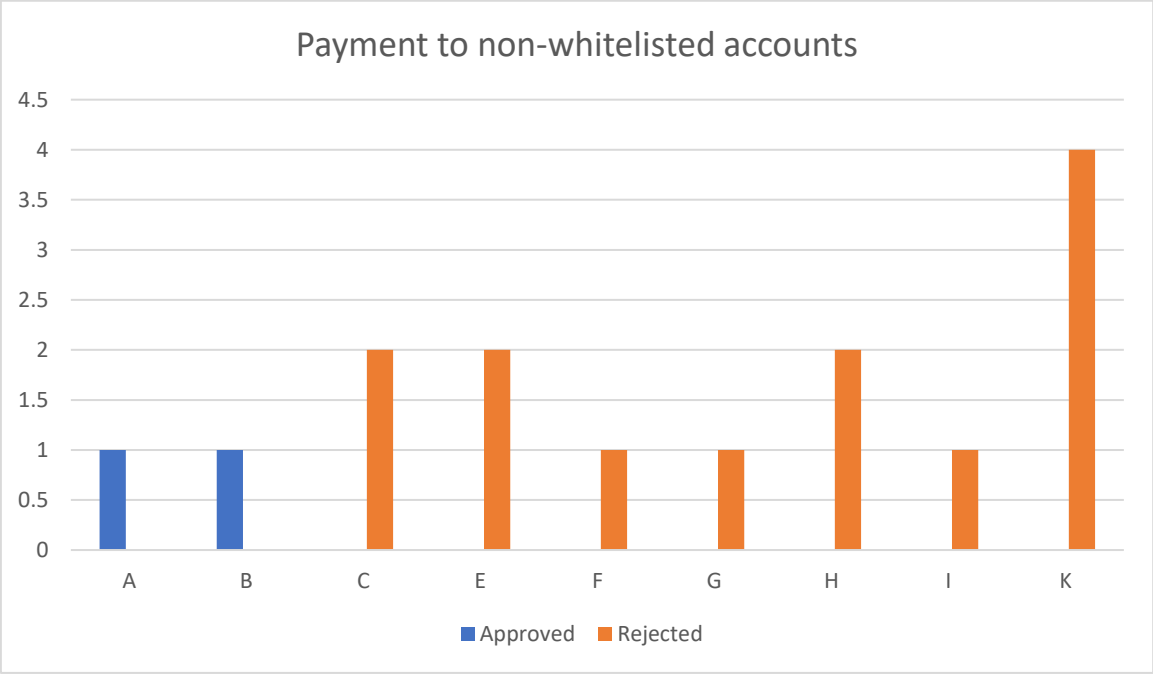

Figure 5.2: Payment to non-whitelisted accounts

5.5.2 Payment from a non-whitelisted account

The transactions in this test also involve a transfer where one of the accounts is not included in the whitelist of accounts permitted to transfer STR tokens. The transfer amounts are within the allowed range of 0-50 STRC.

Recipient Address: Account B

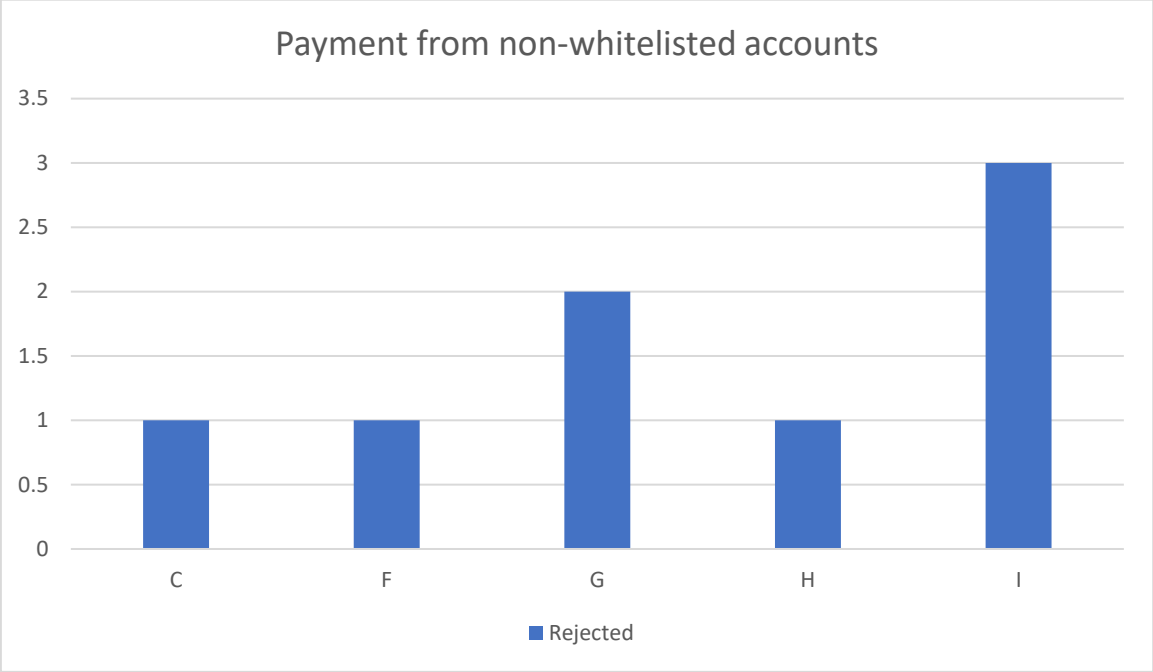

Figure 5.3: Payment from a non-whitelisted account

5.5.3 Transfer assets with insufficient transaction fee

The transaction fee for an asset transfer has been set to 0.001 Algos (shown as 1000 microAlgos).

A list of time series transfer transactions was attempted where this transaction fee was not met. These transactions had varying asset amounts and transaction fees. Although the transaction fees varied, all their values were less than the required transaction fee of 0.001 Algos.

Transacting accounts:

Sender: Account A

Recipient: Account B

Escrow: Account L

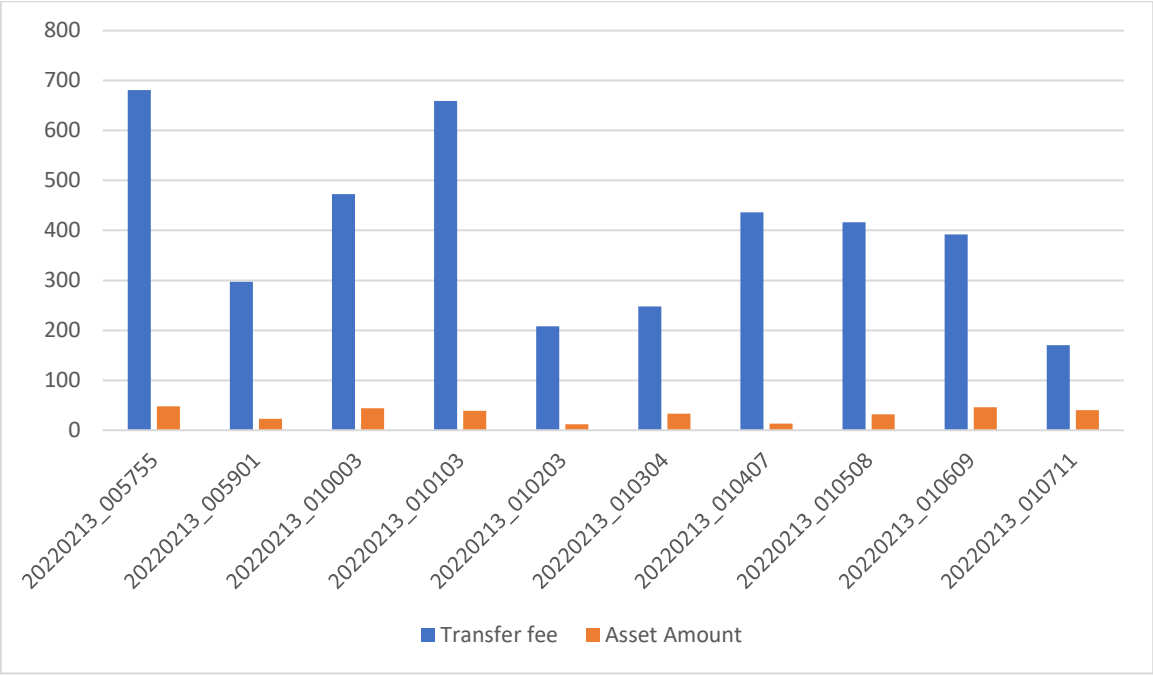

Figure 5.4: Transfer assets with insufficient transaction fee.

5.5.4 Transfer without a transaction fee

The difference between the transaction below and the above (Section 5.5.3) is that the escrow transaction fee exists in the group transaction. The transactions below represent incomplete atomic transactions where the escrow transaction fee does not exist at all in the group transaction.

In this simulation, the transactions were executed from and to the same accounts through the escrow with different payment amounts in intervals of a minute between each other.

Transacting accounts:

Sender: Account A

Recipient: Account B

Escrow: Account L

Table 5.2: Transfer assets without a transaction fee

Index	Timestamp	Transaction ID	Asset Amount	Result
1	20220213_011253	ALIXR5JZPBLAB3H5DHEYK6XP7A5 HIWHYJHQVBI44KT5UHRTV3CTQ	41	Transaction rejected by logic
2	20220213_011354	7MYI4RLFU5YWK7R2GYDHU2SD7 4VNZPS43IZN552FXV5IVA4BBZGQ	8	Transaction rejected by logic
3	20220213_011454	FDZYJJGEQ2F2OO6RJ72K6XIJGCG 6AF7DBRKUTCUMWMT7FYV74NLA	46	Transaction rejected by logic
4	20220213_011554	TRDWTH2QTJMSCUE3NIHPJ6GLLC CUD4BLF6NHFBT6R4BRXKTQVVHA	16	Transaction rejected by logic
5	20220213_011656	77L7BENIFLJET7KPAYSLFAJ4JZWD VGK6KPRBDHRW4CSYCZGEWVHA	49	Transaction rejected by logic
6	20220213_011753	CRJ54D52UAEMNRG3LD36JLCH5TF LA54JPWVFSSNUBPCKDYX44WQQ	28	Transaction rejected by logic
7	20220213_011849	B4PZPE54FGSLPLZTYCCS5C4BZE Z2E26SO2ILJ4V47TXNA3YNKREA	39	Transaction rejected by logic
8	20220213_011945	L6MCCVNZYNMLI525NYVADIT4CW I5SMPCX77TC67BF2HZVIKJZRFQ	3	Transaction rejected by logic
9	20220213_012042	V2ZHH7IBSW4BHDARPTW45RRWAA FH5JGEEFRHFSBMJZ72XZJYQFDA	13	Transaction rejected by logic
10	20220213_012140	7WGMXU2LATYWDGJ25G2O5VF6ME CNQW5EFA2PVMECEP2VNU3TXXWQ	42	Transaction rejected by logic

5.5.5 Transfer STR outside the permitted range

The minimum allowed transfer amount is set to 0 STRC, and the minimum is 50 STRC per transaction. In this simulation, each transaction attempts to breach this limit by attempting to transfer either less than the minimum or more than 50 STRC. A random function was used for each transaction to determine the amount of STRC being transferred. This function generates a number within a range from -50 to 150. Each transaction was executed within intervals of one minute.

Transacting accounts:

Sender: Account A

Recipient: Account B

Escrow: Account L

Table 5.3: Transfer of STR in outside of the permitted range

Index	Timestamp	Transaction ID	Asset Amount	Result
1	20220213_004103	N/A	-26	Error: invalid argument "-26" for "-a, --amount"
2	20220213_004141	N/A	-31	Error: invalid argument "-31" for "-a, --amount"
3	20220213_004219	ID5V7EF3YFBIDGO7D3UQPVQ4CFKXTVC43UXNEWN2GHRZXGLLNZBQ	62	Transaction rejected by ApprovalProgram
4	20220213_004321	N/A	-2	Error: invalid argument "-2" for "-a
5	20220213_004359	SANP6X3UVRUNRJHCVUMBAC4SCLQ2IRWSVYVDTNJVG4MDEJISWLFQ	66	Transaction rejected by ApprovalProgram
6	20220213_004500	N/A	-3	Error: invalid argument "-3" for "-a, --amount"
7	20220213_004538	QHRO4KI2OJVCYAM4DPMYUFYC6Z4DM7KMISWMMK6ZXPFPFEYAX7AA	84	Transaction rejected by logic
8	20220213_004641	N/A	-19	Error: invalid argument "-19" for "-a, --amount"
9	20220213_004719	C6NAUBCVGB2PQ542GGRCP2DAPLMI3LX3IOQKE4PYMYGEVECPBVZQ	23	Success. Transaction committed in round 10620
10	20220213_004928	6RSZ6ZGO4EBVR3PK5E5Y2JRKJBH6RYBKLEJTR7IIXZNLYJOEPSQQ	52	Transaction rejected by ApprovalProgram

5.5.6 Test for success

As part of the simulation, one of the tests were designed to verify that transaction that adhered to the attached conditions would be accepted by the smart contract and written to the blockchain.

A total of 10 transactions were executed as part of the test. Different amounts of STRC were being transferred with each transaction. Both the sender and the recipients were whitelisted and allowed to transfer STRC between each other. The STRC amount for each transfer was within the permitted range of 0-50 STRC per transaction.

Transacting accounts:

Sender: Account A

Recipient: Account B

Escrow: Account L

Figure 5.5: Transfer transactions are not in violation of the contract

5.6 Performance-Based Simulation Result

In this section, various simulations were conducted to gauge how the artefact would fare under realistic conditions through a simulation consisting of a range of tests. This test involves large amounts of requests sent by concurrent simulation users to test the capacity of users that the system can accommodate simultaneously. These requests were asynchronous and were all fired simultaneously with the expectation that all would be processed successfully. Any detected failures contributed to determining the limitations in the number of users and concurrent requests.

5.6.1 Repetition Simulation

In the second simulation, the number of iterations increased by a factor of 150 for each iteration for a total of 100 iterations. The simulation ran through for a total of 1 hour, with each iteration lasting exponentially longer than the previous one. There were 10

concurrent users for each iteration, and all the requests were fired simultaneously in each iteration.

A sample of 10 records was randomly selected from 100 data points collected during the simulation.

5.6.1.1 Data transfer per transaction count

The data transfer rate increased in a ratio which was directly proportional to the increase in the number of requests. This transfer rate was a measurement of the bytes returned from the server and depended on each request's success. The transfer rate increase being consistently proportional to the increase in the number of requests per iteration indicates a successful simulation because a failed output would not transfer data and contribute to the data being inconsistent.

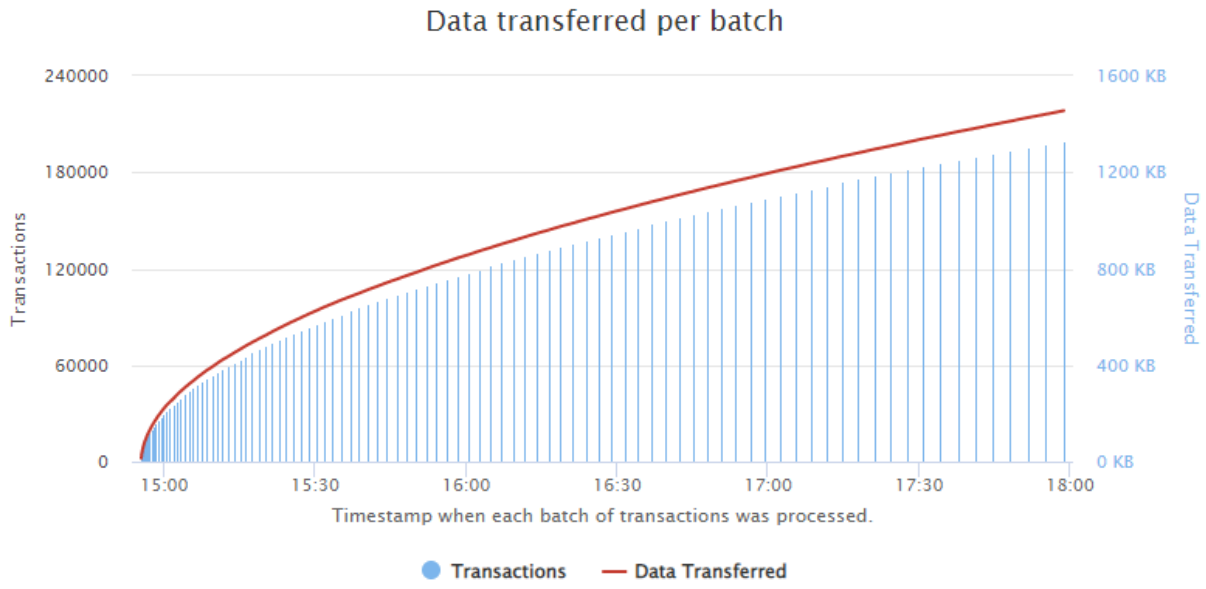

Figure 5.6: Data transferred per batch

As suggested in Figure 5.6, the data represents a linear relationship with a complexity of T(N) = O(N) regarding the ratio of data transferred per request.

5.6.1.2 Transaction rate

The data showed no relationship between the number of records processed per second (transaction rate) and the increasing number of requests in each round. The data showed that the transaction rate was sporadic throughout the simulation and the amount of requests to be processed in each round was not a factor in the amount of records being processed at any given time.

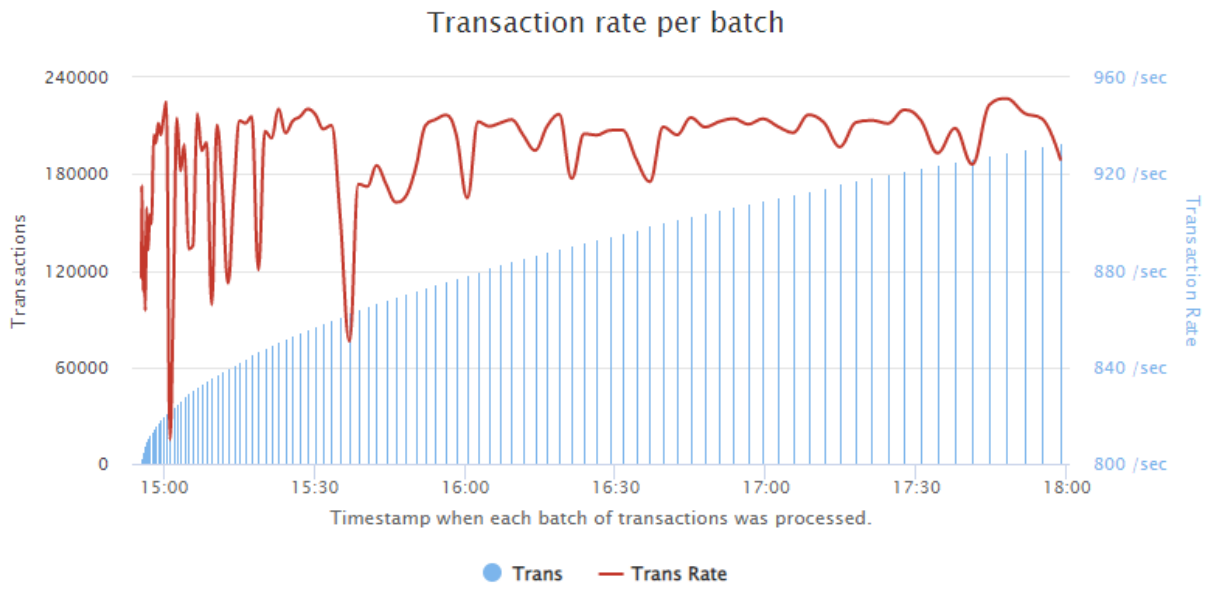

Figure 5.7: Transaction rate per batch

In terms of time complexity, the execution of this simulation revealed an average case scenario of T(n) = O(n), which displays a linear trend.

5.6.1.3 Throughput

Throughout the simulation, the transaction throughput displayed sporadic behaviour, and the trend did not correspond to the increasing amount of transactions per round. Although the data displayed sporadic behaviour, the throughput stayed within the rate of 6 bytes per second.

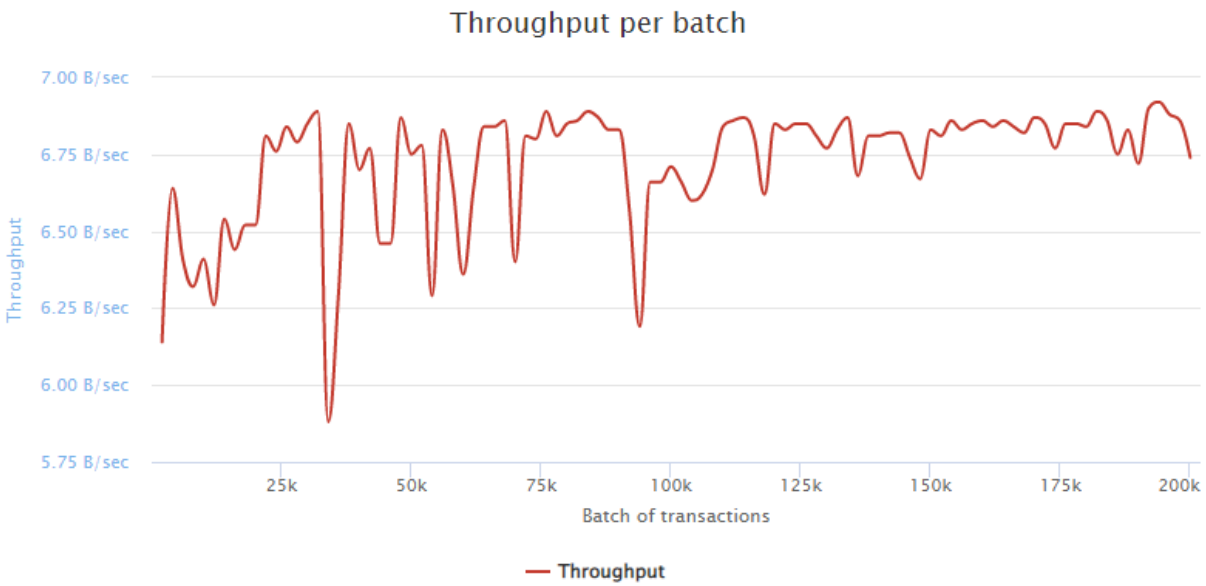

Figure 5.8: Throughput per batch

5.6.2 Concurrency Simulation

The second part of the performance simulation included a test to gauge the performance of the artefact as the number of concurrent users firing requests simultaneously increased. Each of the users fired the same number of requests.

5.6.2.1 Data transfer per batch

Similar to the repetition simulation, the data transferred total maintained a proportional ratio to the increase of the transaction count per batch in each round, showing a space complexity of T(N) = O(N).

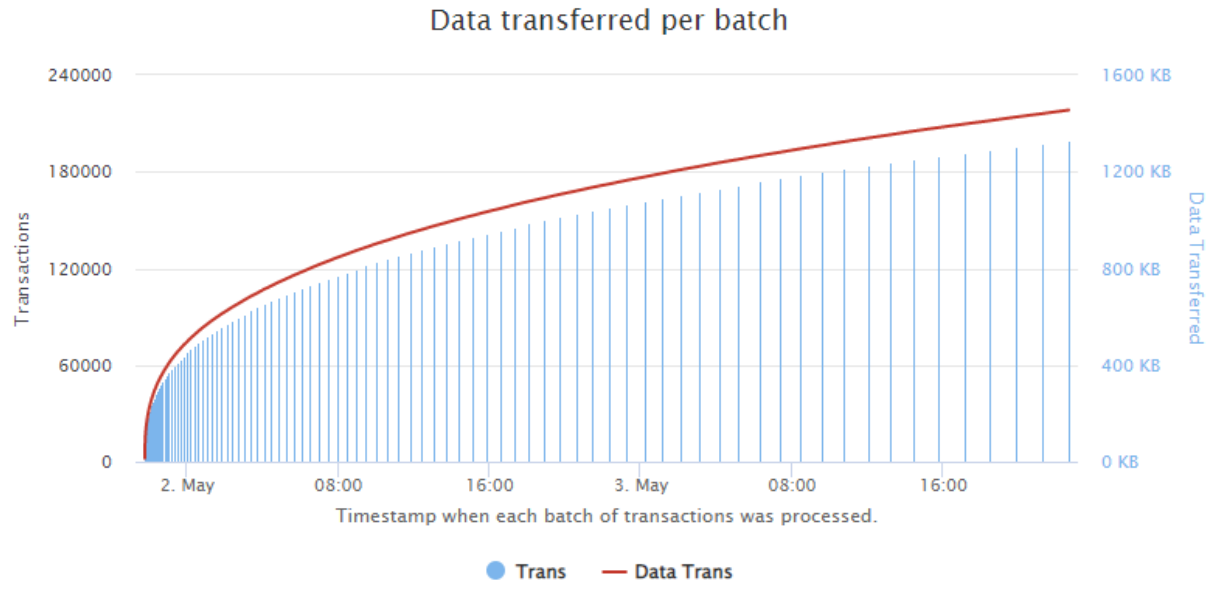

Figure 5.9: Data transferred per batch

5.6.2.2 Transaction rate per batch

For the concurrency simulation, the data showed a sharp decline in the amount of transactions processed as the amount of transactions per batch increased. The trend in Figure 5.10 revealed an inverse relationship between the increasing batches and the transaction rate.

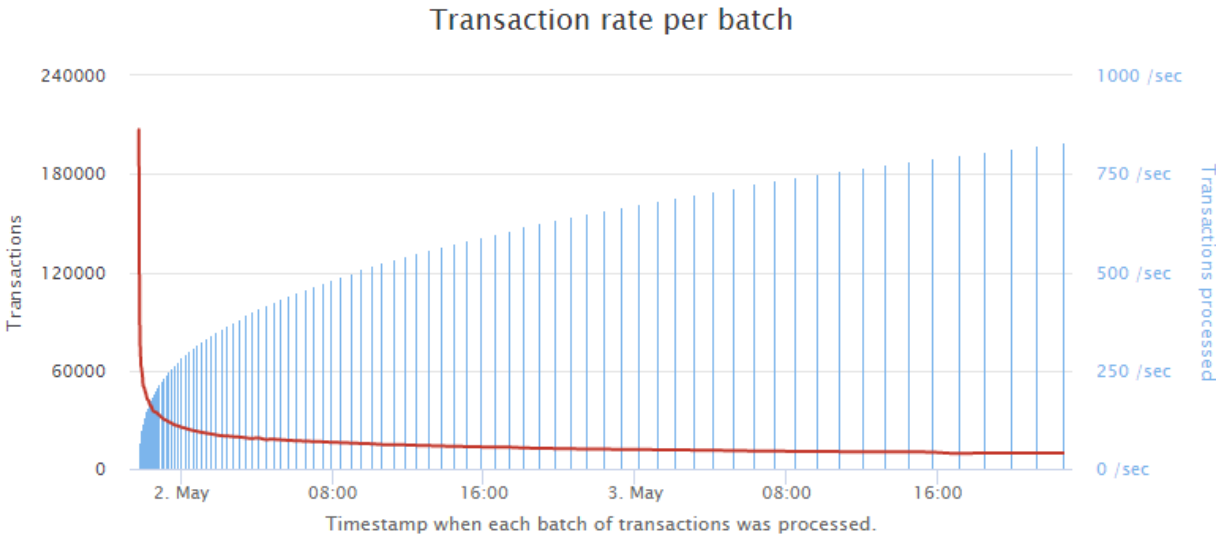

Figure 5.10: Concurrency simulation – Transaction rate per batch

5.6.2.3 Throughput

The throughput displayed an inverse relationship to the amount of transactions. As the amount of transactions increased with each round, there was a clear downward trend, with the throughput decreasing at an exponential rate.

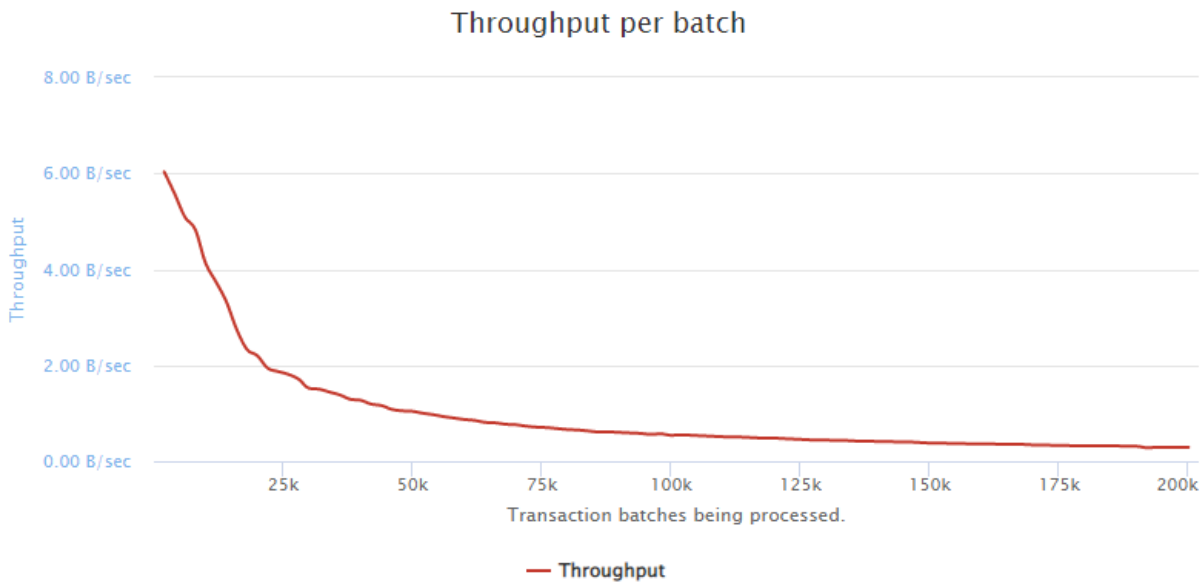

Figure 5.11: Concurrency simulation - Throughput per batch

5.6.3 Multi URL Simulation

The third part of the simulation included firing many simultaneous requests to the artefacts via multiple endpoints to observe any changes in the performance.

The services included a retrieval of account information, a retrieval of asset information, and a retrieval of application (herein referred to as the smart contract) information in the following endpoints:

- /v2/status
- /v2/accounts/${account_id}
- /v2/applications/${app_id}
- /v2/assets/${asset_id}

5.6.3.1 Data transferred per round

There appeared to be less data transferred per round in the simulation for multiple requests compared to the simulation using a single URL though the trend of increasing at a proportional rate to the increase in the number of requests per batch remained the same.

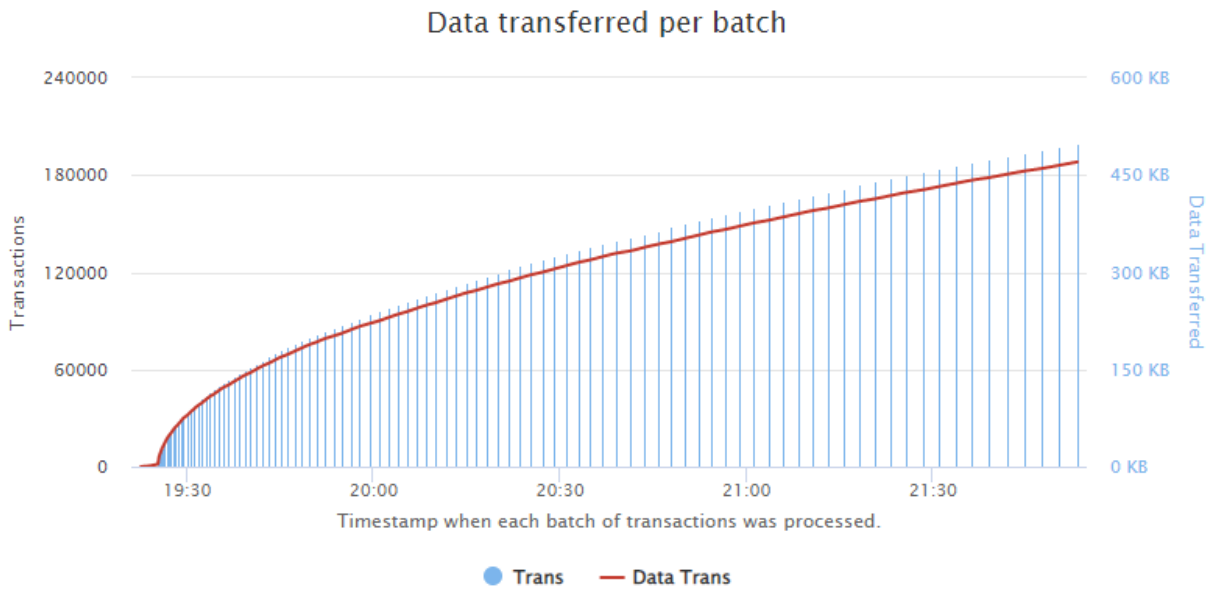

Figure 5.12: MultiURL - Data transferred per round

5.6.3.2 Transaction rate

The data showed a non-existent relationship between the amount of transactions being processed per second and the increase in the amount of transactions being processed. When ignoring a few outliers, the transaction rate maintained an average of 1137.42 transactions per second. Figure 5.12 shows a space complexity of T(n) = O(1).

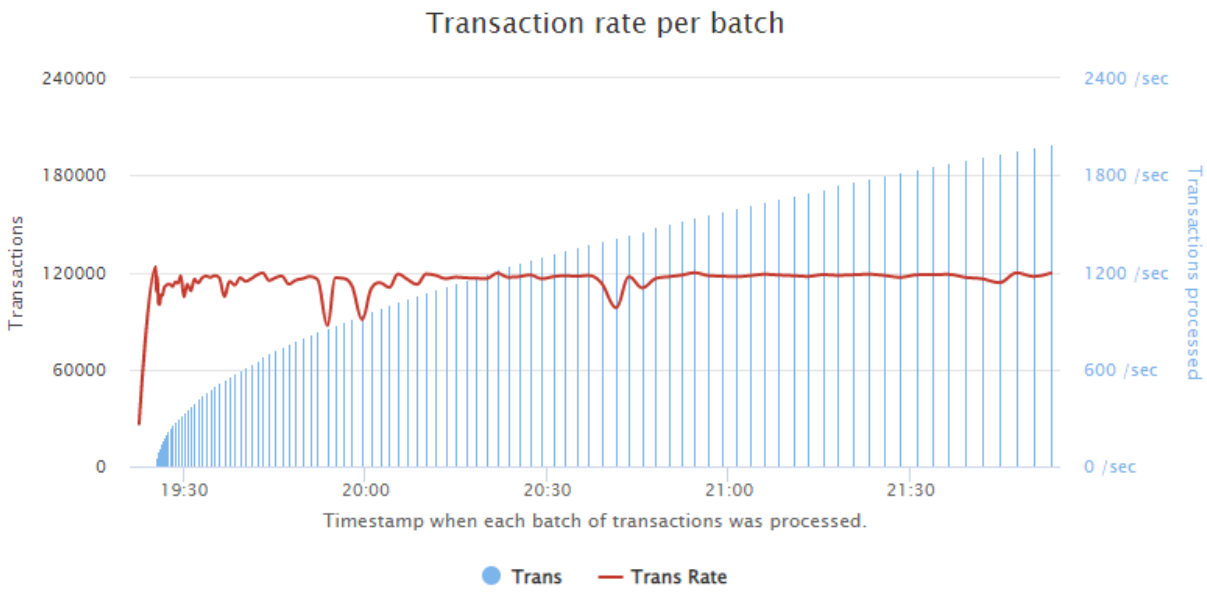

Figure 5.13: Multi-URL – Transaction rate per batch

5.6.3.3 Throughput

With the first data point being an outlier of 0 bytes per second, the throughput maintained a time complexity of T(n) = O(1). The simulation maintained a consistency ranging within 2 bytes per second despite the number of transactions increasing by 2000 hits in each round.

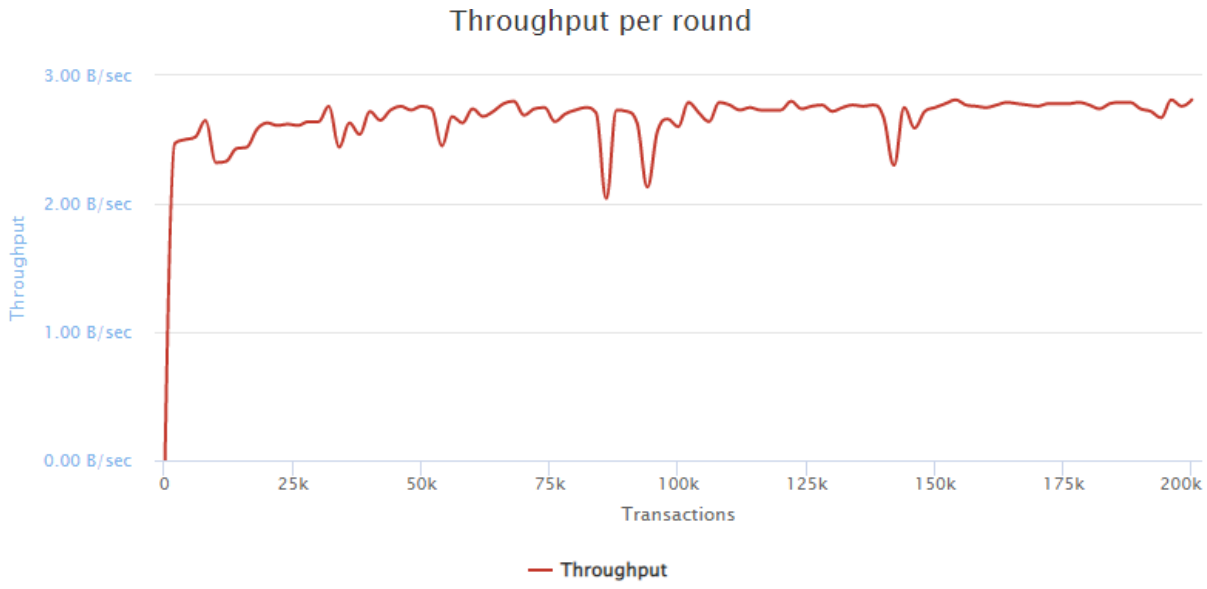

Figure 5.14: Multi-URL - Throughput per round

5.7 Chapter Summary

Chapter five discussed in detail how the simulation was set up and conducted. This is a fulfilment of the fourth (Section 3.2.4) activity of the DSRM processes outlined by Peffers et al. (2006).

Design science research should demonstrate the efficacy of the artefact to solve the problem through various activities involving experimentation, simulation, and case studies (Peffers et al.,2006:90).

This chapter outlined the participating accounts, the tokens used in the transfer, the smart contract that was created, and the Algorand sandbox as the blockchain environment in which the simulation was conducted. It also discussed the different simulation tests where transfers of STR assets with different conditions were subjected to the programmable money environment, and the results of this simulation were recorded.

Additionally, a performance simulation was conducted. High volumes of dummy requests were submitted during this simulation to gauge the artefact's resilience under stressful conditions. In the first part of the simulation, the amount of simultaneous transactions was increased. In the second part, the number of threads was increased. Finally, the number of endpoints was increased from one to four.

CHAPTER SIX : RESEARCH FINDINGS AND DISCUSSION

6.1 Introduction

This chapter discusses the results and findings of the experiment. The results are the output of multiple time series tests with different parameters but subjected to the same artefact to measure how the system behaves under each condition.

6.2 Discussion of results

The discussion of the research study's results is focused on the data collected during the evaluation of how efficiently the developed artefact enforces pre-determined agreements. This is aligned with the main aim and objectives of this research.

6.2.1 Evaluation of transfers to non-whitelist accounts

The first test in the simulation was the whitelist test (Section 5.5.1), where the system displayed a strong performance. System exceptions were thrown in transactions where the recipient account was not opted into the governing smart contract. In transactions where the account address was opted in but not part of the whitelisted accounts, the smart contract rejected the transaction. Transactions whose recipient accounts were whitelisted went through successfully.

Similar to the first test, the second test in the simulation was also a whitelist test (Section 5.5.2). The difference in this test was that it was the sender and not the recipient account, which was not in the whitelist of accounts permitted to transfer STR coins. The system also displayed a strong performance for this test. None of the accounts in each transaction were in the whitelist, and the approval program accordingly rejected each transaction. An unexpected result was from the last transaction (transaction RCNLG4BAKCW32546TOXA2WXWBCTDIGKEGHE6MAIIVOULT5FM5DIA) in which the atomic transaction failed due to the account having insufficient funds to pay the transaction fee.

6.2.2 Evaluation of transfers with insufficient or non-existent transaction fees

The second test in the simulation was a token transfer transaction where the transaction fee supplied was insufficient (Section 5.5.3). Transaction fees in the Algorand blockchain are set to 0.0001 microAlgos (define Algos). In addition, a test was also conducted in which the settlement of the transaction fee was completely excluded from the atomic transaction (Section 5.5.4). A total of 10 transactions were executed, and the output was unanimously rejected by the logic.

6.2.3 Evaluation of transfers where the transfer amount violated the permitted range

The transfer of STR tokens outside the permitted range of 0 – 50 STRC test (Section 5.5.5) revealed that the smart contract application does not allow the transaction to go through if the amount exceeds the maximum of 50 STRC. The system also rejects the transaction as a syntax error when the transfer is a negative value. Transfers of STRC with amounts that fall within the permitted range are successfully recorded to the blockchain.

6.2.4 Evaluation of transactions that do not violate agreements

The final test in the simulation evaluated how the system performed when the terms of the transaction were not violated. A total of 10 transactions were executed, and as expected, they were unanimously approved by the smart contract and successfully written to the blockchain.

6.2.5 Performance evaluation

The artefact was also submitted to a performance simulation where a range of load tests with different parameters was fired simultaneously to benchmark the system performance under stressful conditions.

The performance simulations included tests where the number of simulated requests was increased by 200 per round, while the concurrency remained constant at 25 simulation users. The second simulation was a concurrency test where the number of simulation users was increased by 200 with each round. Each user fired 100 requests, and there were no delays between each request. The third performance simulation

included tests where the number of simulated requests increased by a sum of 200 per round while the concurrency level remained the same. However, in this simulation, 4 URL endpoints were accessed instead of one.

6.2.5.1 Time elapsed comparison

Looking at the time, it took to complete each simulation. It was clear the concurrent user simulation was where the artefact performed at its worst since it took more than 2 days to complete the simulation compared to the requests simulations, which lasted roughly 2 hours even though the sum of requests was exactly the same.

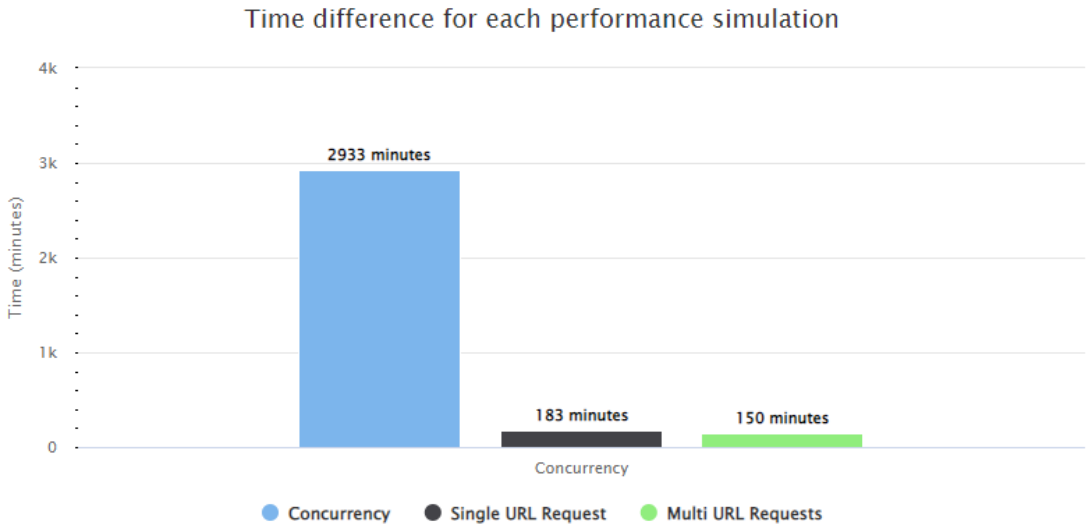

Figure 6.1: Difference in the time elapsed for each test in the performance simulation

6.2.5.2 Concurrency

The concurrency simulation once again had the poorest performance of its counterparts when it came to the execution of simultaneous threads. As seen in Figure 6.2, it took a total of 2 days to complete the concurrency simulation compared to the 2-hour average for the repetition counterparts.

As shown in Figure 6.2, the elapsed time grew by an exponential function as the number of requests produced by the increasing concurrent users increased.

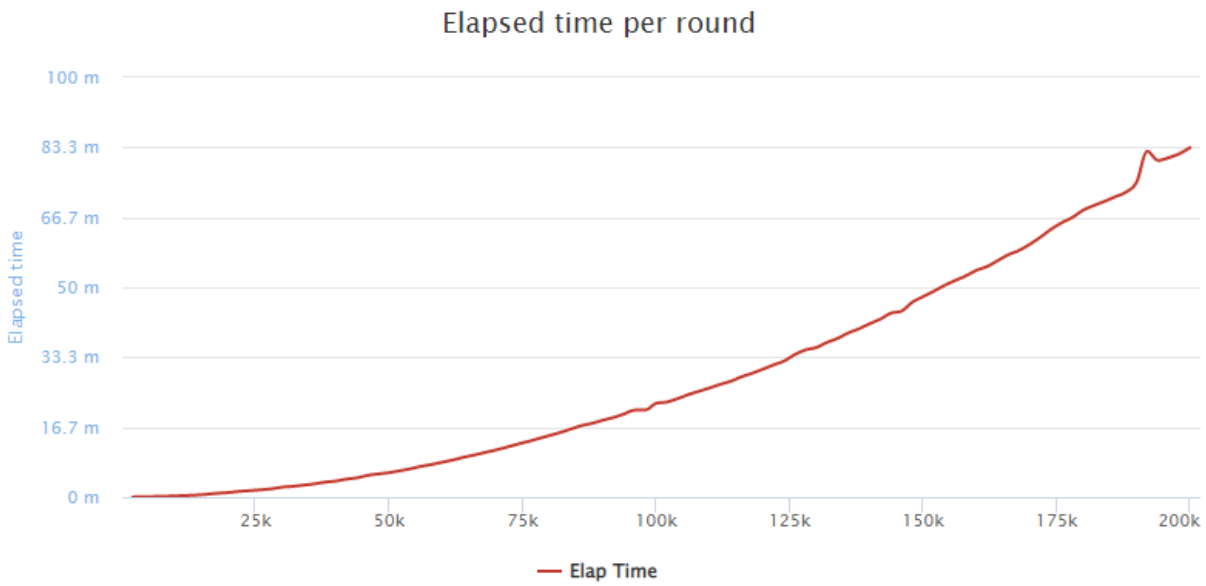

Figure 6.2: Elapsed time per round in the concurrency simulation

6.2.5.3 Time elapsed as the number of requests increases

Although the time complexity of the multiple requests simulation was also increasing at an exponential rate, the performance was significantly better when there were not too many concurrent users.

The data shown in Figure 6.3 displays a linear relationship between the number of requests sent in each round and the time it took to process them.

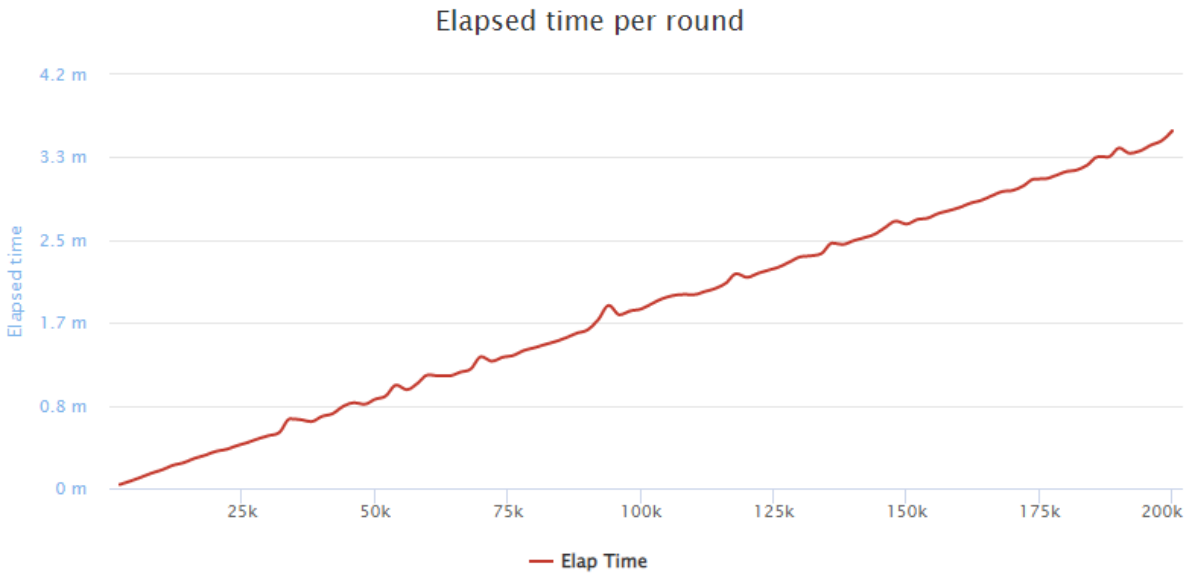

Figure 6.3: Elapsed time per round in the multiple-request simulation

6.2.5.4 Time elapsed as the number of requests increases in multiple URL simulation

The simulation where many requests were fired to multiple URLs displayed the best performance. Even though there were visible spikes, the data presented a linear trend in the time it took to process the bulk requests as they increased within each round.

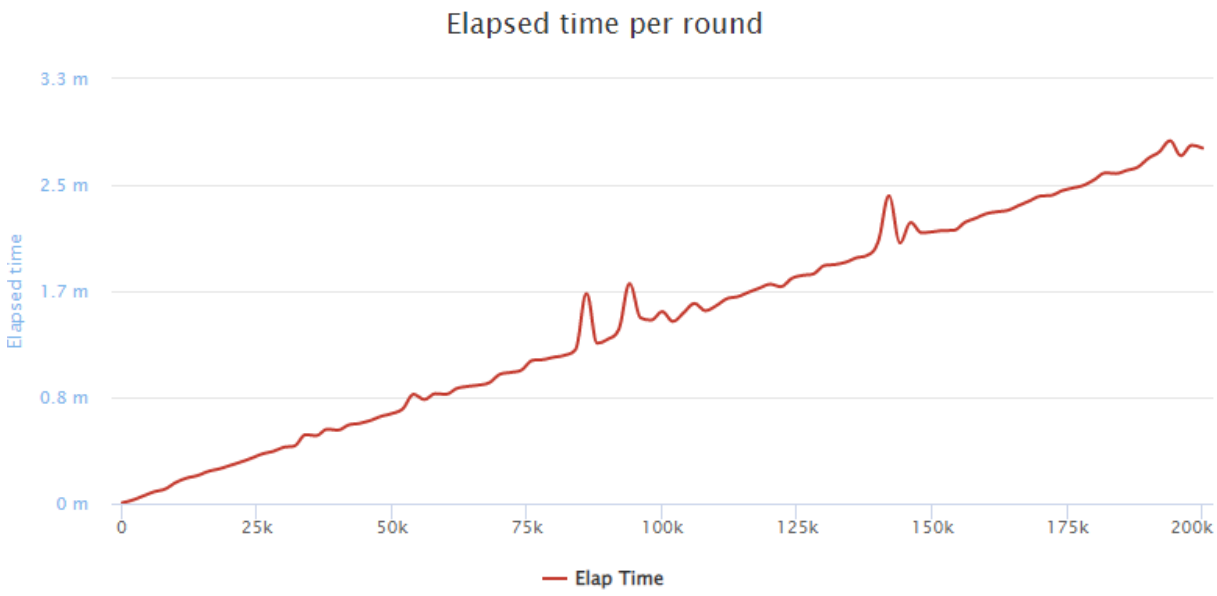

Figure 6.4: Elapsed time per round in the multiple URL's simulation

6.3 Problems encountered and Provided Remedies

6.3.1 Deployment on long-running networks

The Purestake Algorand API service could not be used since the standard free subscription has a limit of 500 000 requests per day, which put a potential restriction on the benchmark simulation tests that required a large number of requests to observe the performance of the artefact under stressful conditions. As a remedy, the simulation was conducted on a local Algorand sandbox, pointing to a private network, and not directly on a long-running network (e.g., BetaNet, TestNet, or MainNet) to address the start-up speed of the sandbox instance. The sandbox is a development environment with Algod and Indexer that runs on a docker container.

6.3.2 Creating dynamic participants during runtime

The first setback during the simulation was a challenge in dynamically creating random accounts to use during the simulation, which limited the randomness of account addresses used in the test. A list of accounts was created upfront and stored in an array to remedy this. A function was used to select a random account to be used in each execution.

6.3.3 Use of cron jobs

The initial plan was to set up a cron job provided by the Ubuntu OS to execute the tests in the simulation. Due to environmental issues where some environment variables needed by the simulation were not available in the cron environment, the tests were not executing successfully. As a remedy, a loop was set up to execute each test in intervals of 30 minutes, achieving the same results expected by cron jobs.

6.3.4 Transaction times

Due to inexperience, the researcher could not include a test with conditions that control the transaction times. An example test would be to limit transactions to business hours and add to the simulation test pack a test that conforms to or violates the said business hours to gauge the artefacts' behaviour under those parameters. However, this test was seen as a nice to have, and the test pack in the simulation was already deemed comprehensive.

6.3.5 Zero value transfers and negative transfers with charges

Algorand uses zero-value transfers to opt into tokens - this is not limited to the first transaction. A discovery was made during the functional tests that zero-value transfers (including transactions with negative amounts) were permitted on the Algorand platform. These transactions were still charged a transaction fee of 0.001 Algos. A potential remedy for such scenarios was to programmatically fail these transactions by implementing a minimum transfer amount on the smart contract.

6.3.6 Feedback on exception handling

Another limitation is on the infrastructure level. The feedback from failed transactions is not intuitive enough. There is no particular description or reason why a transaction failed. Messages on failed transactions are rather generic. An example output is *Transaction rejected by ApprovalProgram* (the approval program is the smart contract responsible for the approval/rejection of transactions before they are committed to the blockchain). This output can be returned for different transactions, each having different reasons for violating the smart contract.

6.3.7 Service endpoints

During the performance testing (Section 5.6), not all service endpoints provided by the artefact were included. This is due to many endpoints and some overheads and constraints that come with the system. One example was that transfer transactions come with a fee of 1000 microAlgos, and running thousands of transfers incurs large transaction fees, and in turn, refunding accounts represented an additional overhead during the end-to-end testing. However, during the functionality simulation testing (Section 5.5), there was an idea of how each individual transfer was supplied.

6.3.8 Evaluation of longevity

The longevity provided by programmable money was not explored. This is a case where programmable money moves to different states through approved transactions. In these different states, the rules change, and the behaviour of the artefact through the smart contract execution is observed. This is a potential candidate for future research.

6.4 Research Findings

A detailed discussion of how the system is designed in both chapter 4 and chapter 5 addresses the research question. Chapter 4 addresses the question through a detailed description of how the programmable money system is designed. Chapter 5 addressed the “enforce usage based on the intent of the allocation” part of the research question through the simulation conducted. In this simulation, NFTs (STR)

with attached agreements are created, participating accounts are also created, and smart contracts are configured and installed into the blockchain resulting in the programmable money system as the artefact. These NFTs are traded through the artefact under different scenarios to gauge the system's performance.

Section 6.2 of this article discusses the results of the evaluation of PM, which took place in the form of a simulation. The evaluation was to determine the efficiency of enforcing rules specified in a contract.

The aim outlined in Section 1.4 of this study reads as follows: How can we build programmable money with characteristics that enable it to enforce usage based on the intent of the allocation?

In relation to this study, analysing the above facts as a whole and relating them to the aim posed at the beginning of the study, summarizing all the results, the following findings can be mapped to the objectives of this study:

Objective 1: *Determine the appropriate blockchain platform to use as a platform.*

A systematic literature review was conducted in chapter 2 to gather data on existing solutions to this research question. Before the systematic literature review, chapter 2 presented a literature review on the various systems that attempt to produce programmable money to some degree.

Section 2.1.2 analyses the different consensus algorithms widely adopted by most well-known blockchains (e.g., Bitcoin, Ethereum). There are a few different types of consensus algorithms that blockchain networks can use to validate transactions and add new blocks to the chain. Some of the more popular algorithms include Proof-of-Work (PoW), Proof-of-Stake (PoS), and Delegated Proof-of-Stake (DPoS).

PoW is the most well-known consensus algorithm and is used by Bitcoin. It is a very secure system, but presented limitations that include scalability issues and computational requirements due to blockchain platforms that used resource-intensive consensus algorithms (Ferdous et al., 2020; Yaga et al., 2018).

PoS is a newer algorithm that is becoming more popular. It is more energy efficient than PoW, but it is also less secure (Yadav and Singh, 2020).

An analysis of smart contracts used by well-known blockchains was also conducted in the systematic literature review (Section 2.4). Various vulnerabilities, from security issues to programming errors, were also picked up in the literature (Grech et al., 2018; Kalra et al., 2018; Perez and Livshits, 2020).

Algorand blockchain was found to have mitigating properties regarding these limitations. Based on the analysis in Section 2.5, Algorand was chosen as the appropriate blockchain for this study.

Objective 2: *Create programmable money based on cryptocurrency tokens on the chosen platform.*

The second research sub-question was formulated to address this objective and read as follows: *How can we create programmable money tokens on the chosen platform?*

PM NFTs were created using the ASA, a cryptocurrency functionality provided by Algorand. Section 4.4.2.4 describes in detail the Algorand Standard Asset technology used to create the PM tokens. Section 4.4.2.5 provides a short description of the token ownership. Thus, the second research sub-question was addressed. Chapter 5 of this study reported on a simulation where these NFTs were being traded.

Objective 3: *Enhance the programmable money based on smart contracts with enforceable rules?*

Smart contract technologies are the cornerstone of the functioning of the solution. A solution can have multiple smart contracts, each with a unique set of rules. These contracts can call each other during different phases of the tokens used within the programmable money system.

Section 4.4.2.2 provides more details on two types of smart contracts, stateful and stateless, while section 2.5 presents the benefits of Algorand smart contracts.

The developed artefact consisted of both types of smart contracts. A clawback escrow account is a stateless smart contract that validates an atomic transaction (Section

5.3.2.2). Based on the content of the atomic transaction, it then makes a call to the permission app, which is a stateful smart contract.

Section 5.3.2.1 presents the rules governing trades on the created cryptocurrency tokens. Section 5.5 reports on an in-depth simulation of the autonomous enforcement of smart contract rules observed in response to simulated transactions. Section 5.6 reported on a non-functional simulation where the artefact was flooded with a large number of requests to gauge how it would perform under stress or realistic conditions, thus addressing the third objective of this research.

Objective 4: *Evaluate the efficiency in the enforcement of these of the above-mentioned rules.*

Chapter 5 of this study reports on a simulation that was undertaken to evaluate the efficiency of enforcing rules by the smart contracts created to enhance the programmable money tokens. This simulation consisted of functional (Section 5.5) and non-functional tests (Section 5.6). The functional tests aimed to evaluate the correctness/accuracy of enforcing pre-determined rules. The non-functional tests were aimed at evaluating the performance of the artefact under stress to determine how well the artefact would perform within realistic conditions (Section 6.2.5).

The result of the simulation was highly successful, with all the functional tests passing as expected. The non-functional tests were mostly successful, except for the concurrency test, where the performance was notably poorer than in other scenarios.

This poor performance can be attributed to the general blockchain infrastructure that has known scalability issues due to requests not being processed by multithreading (used in traditional systems) but rather consensus methods where each node in the network must sequentially validate transactions before they are published to the blockchain (Chauhan et al., 2018).

CHAPTER SEVEN: RESEARCH EVALUATION AND CONCLUSION

The first chapter of this study presented the problem, along with the aim and objectives of this study. These aims and objectives were reiterated in Chapter 4. This chapter presents an assessment of the research process used in this study to achieve the design for programmable money in the context of the aim and objectives presented in the first chapter.

7.1 Hevner's DSR Guidelines

This section outlines the seven guidelines for design science in information systems research proposed by Hevner et al. (2004) and how each guideline maps to what was addressed in this study.

7.1.1 Design as an artefact

In IS, an IT artefact should result from design science research. This is according to Hevner et al. (2004:81). This artefact must be in the "form of a construct, a model, a method, or an instantiation". The artefact is developed to address the problem.

This research resulted in programmable money as a trustworthy virtual currency. Chapter 4 presented the design concept of programmable money on the Algorand blockchain. The smart contract, which is the cornerstone of the enforceability of contracts, was explained along with the associated NFTs that are traded within the bounds of predetermined contracts.

7.1.2 Problem relevance

The second guideline specifies that the IT artefacts described in the first guideline should be solutions to "important and relevant business problems" (Hevner et al., 2004:83).

Sections 1.2 and 1.3 of Chapter 1 provided the background and motivation for this research and stated the research problem that addresses the relevance of this research. The problem space that determines the relevance of a DSR endeavour is described in Chapter 3.

7.1.3 Design Evaluation

Guideline 3 emphasized a thorough evaluation of the designed artefact. “The utility, quality, and efficacy of a designed artefact must be rigorously demonstrated via well-executed evaluation methods” (Hevner et al., 2004:83).

Chapter 5 of this research presented the evaluation of programmable money through a simulation. There were two parts to the simulation, the functional simulation and the non-functional simulation. Each part consisted of multiple rigorous tests.

In the functional simulation, the artefact was subjected to a range of different scenarios to gauge the outcome of transactions under certain conditions. The non-functional test was primarily stress testing the artefact that included tests where the artefact was flooded with a high volume of simultaneous requests and tests where the artefact was subjected to a high volume of concurrent dummy users firing a high volume of simultaneous requests.

7.1.4 Research Contributions

Guideline 4 states that DSR must provide contributions using the designed artefact, design foundations, and/or design methodologies.

The key contributions of this research can be mapped directly to the objectives that have been achieved:

Practical contributions

Build a cryptocurrency platform that can issue tokens

Programmable money was developed on the Algorand blockchain in a local sandbox where multiple accounts and applications were set up in preparation for a proof-of-concept simulation reported in chapter 5 of this paper. This aligns with the first objective of this research.

Build tokens with enforceable rules that run on the platform mentioned above

Using Algorand Standard Assets (ASA), STR tokens were created with attached agreements (section 4.4.2.4). These tokens were used as the medium of exchange during the simulation between the accounts mentioned earlier.

Create smart contracts to manage the rules mentioned above

Each transaction had to go through two main smart contract applications during the simulation. A gatekeeper smart contract that validates each request and ensures it makes a call to a permission smart contract that handles the enforcement of agreements.

Create a platform that allows market players to use these tokens and assess how efficiently these tokens enforce the rules.

A simulation was conducted on the Algorand blockchain in a private network. The simulation is reported in chapter 5. This simulation included functional and non-functional tests. The functional tests were mainly an evaluation of the enforcement of contracts. The non-functional tests focused on benchmarking the platform's performance under high loads. The simulation was mostly successful, and the results were discussed in chapter 6 of this research.

Additional contributions can be found in Section 1.8.2 of Chapter 1.

7.1.5 Research Rigor

Guideline 5 emphasizes the application of rigorous methods in the construction and evaluation of the design artefact.

In Chapter 3, the research methodology was presented. This included the research paradigm, methodology, methods, and techniques based on the Pyramid of Jonker and Pennink (2009). Chapter 3 also explains the flow of the research guided by the Oates model (2012). Chapter 3 also introduces design science research which is the cornerstone of this research. Hevner’s (2004) seven DSR Guidelines are introduced, and the DSR activities followed by this research are articulated.

7.1.6 Design as a Search Process

Guideline 6 embodies the information systems research framework where the researcher consults the knowledge base to maximise rigour and applies the IT artefact in the appropriate environment during the development and evaluation activities in the design cycle (Hevner et al., 2004:83).

Chapter 5 of this research presented the evaluation of programmable money through a simulation. This section reports on a simulation where the programmable money

artefact was intensely evaluated against the objectives stated in section 1.4 of Chapter 1. Other solutions in this area were explored through a systematic literature review presented in Chapter 2.

7.1.7 Communication of Research

Guideline 7 emphasizes the effective communication of design-science research to both technology-oriented and management-oriented audiences.

The research was communicated throughout the lifecycle of the research with various stakeholders, including the supervisor, the Faculty Research Committee (FRC), other supervisors and fellow students during research symposiums, and members of the supervisors' research group during research meetings.

This research will also be communicated in the form of a thesis and a published article.

7.2 Conclusion

This thesis reports on a design science research project regarding the design of programmable money as a trustworthy virtual currency. The inspiration behind this thesis is the need to enforce contracts and agreements when conducting financial transactions – which relates to the problem being addressed by the study (section 1.3). This need arises due to the lack of properties in the current currency, such as the ability to attach policies and agreements, the longevity to track the movement of money through multiple transactions, and the capability of making anonymous decisions.

To make this possible, this research proposed a solution that embeds the behaviour in the money rather than the person. The research aims to develop a virtual currency that can be trusted to enforce hardwired terms and conditions during the execution of a transaction (Section 1.4). Once the contracts and agreements are finalized, they are attached to the currency. In turn, the power and the decision-making ability are in possession of the currency and not the human.

The research project included a literature review which was conducted to align this research in view of previous studies, and in addition, a systematic literature review was also conducted. A systematic literature review gathered knowledge from past

papers regarding existing solutions on trusted currencies capable of enforcing contracts. These existing solutions were compared to one another and used as input to the design of the proposed artefact based on the quality of these papers.

The design of the solution is centred around blockchain technologies. It encompasses a combination of blockchain solutions that include NFTs, smart contracts, and atomic transfers. An NFT called STR (Stable Rand) was created along with an associated smart contract that defines the conditions under which the NFT can be transferred. These conditions range from allowable transfer amounts to whitelists of accounts permitted to transact on the NFT.

The developed artefact was evaluated through a simulation on a docker sandbox installed in an Ubuntu Virtual machine. In this sandbox, the Algorand blockchain was configured. The simulation consisted of functional and non-functional tests.

The functional test pack aimed to evaluate how the programmable money system behaves in scenarios where agreements associated with currency being traded (STRC) are either violated or conformed to. The non-functional test pack aimed to observe how the artefact would perform under more realistic conditions where many users send transactions to be processed simultaneously. The input data was largely based on random data, while the tests in the simulation were based on numerous requests fired at short intervals. These rapid tests consisting of randomized data helped simulate real-life human interaction with significantly higher frequency.

Analysis of time series data collected from the simulation results proved the simulation very successful for the functional tests. The artefact was largely successful in processing a large number of requests, with each request processed successfully. There was a significantly poorer performance observed from a concurrency simulation where many simulation users sent simultaneous requests. The contributing factor is the fact that the concept of concurrency is not currently supported in blockchain due to the consensus methods.

Algorand’s consensus method is proof of stake and not proof of work which improves the processing time per node but still does not deal with the scalability faced by blockchains in general. This scalability issue poses a question of how well the artefact can serve in a real-world scenario while running within the blockchain infrastructure

and whether that comes with blockchain properties, making the scalability challenge a worthy trade-off.

7.3 Future Work

The concept of programmable money is complex and presents a range of potential future work. For example, work will also need to be done to ensure that programmable money is secure and reliable. We could also research how programmable money could be used to support new economic models, such as peer-to-peer lending or micro-payments.

Even though this research focused on programmable money in a generic sense, the scope of the research project was centred around financial transactions. The focus on future work can extend to different social contexts, such as law – where the legalities of smart contracts can be explored. Education – where the governance of bursaries could be improved, Other social contexts include supply chain management, politics, health and many more.

The features of the artefact can also be enriched to include other available functionality offered by Algorand, an example of which is the Multisig functionality. This functionality mandates a minimum count of signatures for a transaction before it is approved and written to the network. These features are available in the Algorand network and present the possibility for an enhanced version of the artefact.

REFERENCES

Agarwal, P., Jalan, S. & Mustafi, A. 2018. 'Decentralized and financial approach to effective charity', in 2018 International Conference on Soft-computing and Network Security (ICSNS). 2018 International Conference on Soft-computing and Network Security (ICSNS), Coimbatore: IEEE, pp. 1–3. doi:10.1109/ICSNS.2018.8573644.

Agarwal, P., Jalan, S. & Mustafi, A. 2018. 'Decentralized and financial approach to effective charity', in 2018 International Conference on Soft-computing and Network Security (ICSNS). 2018 International Conference on Soft-computing and Network Security (ICSNS), Coimbatore: IEEE, pp. 1–3. doi:10.1109/ICSNS.2018.8573644.

Algorand. n.d. *Algorand Features & Capabilities in Layer-1*. https://www.algorand.com/technology [Accessed: 9 October 2021].

Agarwal, P., Jalan, S., & Mustafi, A., 2018. Decentralized and financial approach to effective charity, In 2018 *International Conference on Soft-Computing and Network Security (ICSNS). Presented at the 2018 International Conference on Soft-computing and Network Security (ICSNS)*, IEEE, Coimbatore, pp. 1–3. https://doi.org/10.1109/ICSNS.2018.8573644

Algorand. 2019. Algorand & ISDA: *Efficiency and Market Opportunity through Asset Tokenization and Programmatic Money*. https://www.algorand.com/resources/blog/algorand-and-isda-asset-tokenization-and-programmatic-money. [10 July 2020].

Bartoletti, M., Bracciali, A., Lepore, C., Scalas, A. and Zunino, R., 2021, March. A formal model of Algorand smart contracts. In International Conference on Financial Cryptography and Data Security (pp. 93-114). Springer, Berlin, Heidelberg.

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

Chen, J. and Micali, S., 2016. Algorand. In arXiv preprint arXiv:1607.01341. pp. 1-4. Available at: https://arxiv.org/abs/1607.01341. [12 July 2021].

Croker, R.A., 2009. An introduction to qualitative research. In Qualitative research in applied linguistics (pp. 3-24). Palgrave Macmillan, London.

Cunha, P.R., Melo, P. and Sebastião, H. 2021. 'From Bitcoin to Central Bank Digital Currencies: Making Sense of the Digital Money Revolution', In *Future Internet*, p. 34.

Developer.algorand.org. 2021. *Guidelines - Algorand Developer Portal*. https://developer.algorand.org/docs/reference/teal/guidelines/ [26 September 2021].

Developer.algorand.org. 2021. *Smart Contract Overview - Algorand Developer Portal.* https://developer.algorand.org/docs/features/asc1/ [21 September 2021].

Elsden C., Trotter L, Harding M, Davies N, Speed C., & Vines J., 2019. Programmable Donations: Exploring Escrow-based Conditional Giving. In *CHI Conference on Human Factors in Computing Systems Proceedings (CHI 2019)*, May 4–9, 2019, Glasgow, Scotland, UK. ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3290605.3300609

Etikan I, Bala K (2017) Sampling and Sampling Methods. Biom Biostat Int J 5(6): 00149. DOI: 10.15406/bbij.2017.05.00149

Fan, W. *et al.* (2020) 'Blockchain-based Distributed Banking for Permissioned and Accountable Financial Transaction Processing', in *2020 29th International Conference on Computer Communications and Networks (ICCCN)*. Honolulu, HI, USA: IEEE, pp. 1–9. doi:10.1109/ICCCN49398.2020.9209687.

Foroglou, G., & Tsilidou, A. 2015. Further applications of the blockchain. *Conference: 12th Student Conference on Managerial Science and Technology, At Athens*. 1, p. 8. https://www.researchgate.net/profile/Georgios_Foroglou/publication/276304843_F

urther_applications_of_the_blockchain/links/5556f20608ae6fd2d8237a34/Further-applications-of-the-blockchain.pdf. [14 October 2017].

Fowler, S.B. and Lapp, V., 2019. Sample size in quantitative research: Sample size will affect the significance of your research. American Nurse Today, 14(5), pp.61-63.

Fulmer J. 2012. 'Concurrency and the Single Siege'. Available at: https://www.joedog.org/2012/02/17/concurrency-single-siege/

Fustino, R. 2020. *Access BetaNet Network using your own node and goal*. Available at: https://developer.algorand.org/tutorials/betanet-goal/ (Accessed: 9 October 2021).

Gaži, P., Kiayias, A. and Zindros, D., 2019, May. Proof-of-stake sidechains. In 2019 IEEE Symposium on Security and Privacy (SP) (pp. 139-156). IEEE.

Gladden, M. 2015. 'Cryptocurrency with a Conscience: Using Artificial Intelligence to Develop Money that Advances Human Ethical Values', Annales. Etyka w życiu gospodarczym, 18(4), p. 90-92. DOI: 10.18778/1899-2226.18.4.06, http://www.annalesonline.uni.lodz.pl/archiwum/2015/2015_4_gladden_85_98.pdf. [20 October 2017].

Göran G. 2012. Pragmatism vs interpretivism in qualitative information systems research, European Journal of Information Systems, (21), 2, 135-146. http://dx.doi.org/10.1057/ejis.2011.54

Grech, N., Kong, M., Jurisevic, A., Brent, L., Scholz, B., Smaragdakis, Y., 2018. MadMax: surviving out-of-gas conditions in Ethereum smart contracts. Proc. ACM Program. Lang. 2, 1–27. https://doi.org/10.1145/3276486

Hevner, A.R. & Chatterjee, S. 2010. Design research in information systems. *New York: Springer Science+Business Media.* 4-5. http://desrist.org/desrist/content/design-science-research-in-information-systems.pdf. [15 November 2019].

Hyvärinen, H., Risius, M., & Friis, G. 2017. A Blockchain-Based Approach Towards Overcoming Financial Fraud in Public Sector Services. Bus Inf Syst Eng 59, 441–456 https://doi.org/10.1007/s12599-017-0502-4. [Accessed 26 May 2020].

Jonker, J. & Pennink, B. 2009. *The Essence of Research Methodology*. Berlin, Heidelberg: Springer Berlin Heidelberg. http://link.springer.com/10.1007/978-3-540-71659-4.

Kim, H. & Laskowski, M. 2018. Toward an ontology-driven blockchain design for supply-chain provenance. Intell Sys Acc Fin Mgmt, [Online]. 25, 18–27. Available at: DOI: 10.1002/isaf.1424. [Accessed 14 August 2019].

Kiviat, T.I., 2015. Beyond bitcoin: Issues in regulating blockchain Transactions. Duke LJ, 65, p.569.

Kofod-Petersen, A., 2012. How to do a structured literature review in computer science. Ver. 0.1. [Accessed 31 July 2020].

Korpela, K., Hallikas, J. & Dahlberg, T. 2017. Digital Supply Chain Transformation toward Blockchain Integration. HICSS., 4182-4183. Available at: http://128.171.57.22/handle/10125/41666. [Accessed 31 July 2020].

Monzo. 2018. Automate your financial life with Monzo and IFTTT. https://monzo.com/blog/2018/06/07/monzo-on-ifttt/. [Accessed 14 March 2020].

Okoli, C., 2015. A guide to conducting a standalone systematic literature review. *Communications of the Association for Information Systems*, 37(1), p.43.

Omohundro, S. 2014. 'Cryptocurrencies, smart contracts, and artificial intelligence', AI Matters, Vol. 1, No. 2, pp.19–21. Available at: DOI: 10.1145/2685328.2685334. [Accessed 16 April 2020].

Omran, Y., Henke, M., Heines, R. & Hofmann, E. 2017. "Blockchain-driven supply chain finance: Towards a conceptual framework from a buyer perspective.". [Online]. Available at: https://www.alexandria.unisg.ch/251095. [Accessed 24 August 2019].

Pandey, S., Goel, S., Bansla, S. & Pandey, D., 2019, March. Crowdfunding fraud prevention using blockchain. In *2019 6th International Conference on Computing for Sustainable Global Development (INDIACom)* (pp. 1028-1034). IEEE., p. 7.

Perez, D. and Livshits, B., 2021. Smart contract vulnerabilities: Vulnerable does not imply exploited. In 30th USENIX Security Symposium (USENIX Security 21) (pp. 1325-1341).

Pfeffers, K., Tuunanen, T., Gengler, C.E., Rossi, M., Hui, W., Virtanen, V. & Bragge, J., 2006, February. The design science research process: A model for producing and presenting information systems research. In *Proceedings of the First International Conference on Design Science Research in Information Systems and Technology (DESRIST 2006), Claremont, CA, USA (pp. 83-106)*., p. 25.

Poelstra, A., 2014. Distributed consensus from proof of stake is impossible. Self-published Paper.

Singh, S., Singh, N., 2016. Blockchain: Future of financial and cyber security, in: *2016 2nd International Conference on Contemporary Computing and Informatics (IC3I)*. Presented at the 2016 2nd International Conference on Contemporary Computing and Informatics (IC3I), pp. 463–467. https://doi.org/10.1109/IC3I.2016.7918009

Tallyn, E., Pschetz, L., Gianni, R., Speed, C. and Elsden, C., 2018. Exploring Machine Autonomy and Provenance Data in Coffee Consumption. *Proceedings of the ACM on Human-Computer Interaction*, 2(CSCW), p.170. https://doi.org/10.1145/3274439. [Accessed 14 March 2020].

Tseng, C.-T., Shang, S.S.C., 2021. Exploring the Sustainability of the Intermediary Role in Blockchain. *Sustainability* 13, 1936. https://doi.org/10.3390/su13041936.

Rowley, J. and Slack, F., 2004. Conducting a literature review. *Management research new*s. P. 131. https://doi.org/10.1108/01409170410784185. [Accessed 10 March 2022].

Vaishnavi, V., Kuechler, W. & Petter, S., 2019. Design science research in information systems. Available at: http://desrist.org/desrist/content/design-science-research-in-information-systems.pdf. `[Accessed 15 November 2019].

Wang, S., Tang, X., Zhang, Y., Chen, J., 2019. Auditable Protocols for Fair Payment and Physical Asset Delivery Based on Smart Contracts. *IEEE Access* 7, 109439–109453. https://doi.org/10.1109/ACCESS.2019.2933860

Weber, I., & Staples, M., Programmable money: Next-generation conditional payments using blockchain–keynote paper. *In Int. Conf. on Cloud Computing and Services Science (CLOSER)(Apr 2021).*

Yadav, A.K. and Singh, K., 2020. Comparative analysis of consensus algorithms of blockchain technology. In Ambient communications and computer systems (pp. 205-218). Springer, Singapore., p. 211.

Yaga, D. *et al.* (2018) *Blockchain technology overview*. NIST IR 8202. Gaithersburg, MD: National Institute of Standards and Technology, p. NIST IR 8202. Available at: https://doi.org/10.6028/NIST.IR.8202.

Yilmaz, K., 2013. Comparison of quantitative and qualitative research traditions: Epistemological, theoretical, and methodological differences. *European journal of education*, 48(2), pp.311-325.

Zaremba, R. 2021. ‘Securities and Permissioned Tokens’. Available at: https://developer.algorand.org/solutions/securities-and-permissioned-tokens/ .

APPENDICES

APPENDIX A: Create Smart Contract Application and Token bash script

```
#!/bin/bash
# creates and updates the app
date '+keyreg-teal-test start %Y%m%d_%H%M%S'

set -e
set -x
set -o pipefail
export SHELLOPTS

WALLET=$1
# Directory of this bash program
DIR="$( cd "$( dirname "${BASH_SOURCE[0]}" )" >/dev/null 2>&1 && pwd )"

gcmd="./sandbox goal"
gcmd2="./sandbox goal -d ../Primary"

# Get one account from each node
ACCOUNT=$(${gcmd} account list|awk '{ print $2 }'|head -n 1)
ACCOUNT2=$(${gcmd2} account list|awk '{ print $2 }'|head -n 1)

echo "Account1="$ACCOUNT
echo "ACCOUNT2="$ACCOUNT2

echo "Copy teal files from host to sandbox"
./sandbox copyTo ./poc.teal
./sandbox copyTo ./poc-clear.teal
./sandbox copyTo ./clawback-escrow.teal

# create test asset in account
ASSETID=$(${gcmd} asset create --creator ${ACCOUNT} --total 10000000 --name STR --unitname STRC --decimals 0 --defaultfrozen=true | awk '{ print $6 }'|tail -n 1)
ASSETID="${ASSETID//$'\r'/}"
echo "Asset ID"=$ASSETID

# account 2 opt into asset
${gcmd2} asset send -a 0 -f ${ACCOUNT2} -t ${ACCOUNT2}  --creator ${ACCOUNT} --assetid ${ASSETID}

# create white list permission appp
APPID=$(${gcmd} app create --creator ${ACCOUNT} --app-arg "int:"${ASSETID} --app-arg "int:2" --approval-prog ./poc.teal --global-byteslices 1 --global-ints 2 --local-byteslices 0 --local-ints 1
--clear-prog ./poc-clear.teal | grep Created | awk '{ print $6 }')
APPID="${APPID//$'\r'/}"
echo "App ID="$APPID

# compile and fund the escrow
ESCROW=$(${gcmd} clerk compile clawback-escrow.teal | awk '{ print $2 }'|tail -n 1)
ESCROW="${ESCROW//$'\r'/}"
echo "Escrow Address="$ESCROW
${gcmd} asset config  --manager ${ACCOUNT} --new-clawback ${ESCROW} --assetid ${ASSETID}

#now lock the asset by clearing the manager
${gcmd} asset info --assetid=${ASSETID}

#optin two accounts
${gcmd} app optin  --app-id $APPID  --from $ACCOUNT
${gcmd2} app optin  --app-id $APPID --from $ACCOUNT2

${gcmd} app read --app-id $APPID --guess-format --global --from $ACCOUNT
```

APPENDIX B: Add whitelist account bash script

```
#!/bin/bash
# creates and updates the app
date '+keyreg-teal-test start %Y%m%d_%H%M%S'

set -e
set -x
set -o pipefail
export SHELLOPTS

WALLET=$1

# Directory of this bash program
DIR="$( cd "$( dirname "${BASH_SOURCE[0]}" )" >/dev/null 2>&1 && pwd )"

gcmd="./sandbox goal"
gcmd2="./sandbox goal -d ../Primary"

# Get one account from each node
ACCOUNT=$(${gcmd} account list|awk '{ print $2 }'|head -n 1)
ACCOUNT2=$(${gcmd2} account list|awk '{ print $2 }'|head -n 1)

echo "Account1="$ACCOUNT
echo "ACCOUNT2="$ACCOUNT2

${gcmd} app call --app-id 32 --app-account=${ACCOUNT} --app-arg "str:add-whitelist" --app-arg "int:1" --from=$ACCOUNT
${gcmd} app call --app-id 32 --app-account=${ACCOUNT2} --app-arg "str:add-whitelist" --app-arg "int:1" --from=$ACCOUNT
```

APPENDIX C: Transfer Bash Script

```bash
#!/bin/bash

date '+keyreg-teal-test start %Y%m%d_%H%M%S'

set -e
set -x
set -o pipefail
export SHELLOPTS

# Directory of this bash program
DIR="$( cd "$( dirname "${BASH_SOURCE[0]}" )" >/dev/null 2>&1 && pwd )"

#gcmd="../../goal -d ../test/Primary"
#gcmd2="../../goal -d ../test/Node"

sandbox="./sandbox"
gcmd="./sandbox goal"
gcmd2="./sandbox goal -d ../Primary"

# Get one account from each node
ACCOUNT=$($gcmd account list|awk '{ print $2 }'|head -n 1)
ACCOUNT2=$($gcmd2 account list|awk '{ print $2 }'|head -n 1)
ESCROW=$($gcmd clerk compile clawback-escrow.teal | awk '{ print $2 }'|tail -n 1)
ESCROW="${ESCROW//$'\r'/}"
ASSET_AMOUNT=$((RANDOM % 51))

echo "Account1="$ACCOUNT
echo "ACCOUNT2="$ACCOUNT2
echo "ESCROW="$ESCROW
echo "ASSET_AMOUNT="$ASSET_AMOUNT

$gcmd app call --app-id 32  --app-arg "str:transfer" --app-account ${ACCOUNT2} --from $ACCOUNT --out=unsginedtransaction1.tx  # smart contract call
$gcmd asset send -a $ASSET_AMOUNT --assetid 30 -f ${ACCOUNT} -t ${ACCOUNT2} --clawback ${ESCROW}  --out=unsginedtransaction2.tx  # transfer from account 1 to account 2 using clawback account (ESCROW)
$gcmd clerk send --from=$ACCOUNT --to=${ESCROW} --amount=1000 --out=unsginedtransaction3.tx # Pay the ESCROW account for the transfer transaction

#make output transactions accessible outside sandbox
${sandbox} copyFrom unsginedtransaction1.tx
${sandbox} copyFrom unsginedtransaction2.tx
${sandbox} copyFrom unsginedtransaction3.tx

cat unsginedtransaction1.tx unsginedtransaction2.tx unsginedtransaction3.tx > combinedtransactions.tx
${sandbox} copyTo combinedtransactions.tx
${gcmd} clerk group -i combinedtransactions.tx -o groupedtransactions.tx
${sandbox} copyFrom groupedtransactions.tx
${gcmd} clerk split -i groupedtransactions.tx -o split.tx
${gcmd} clerk sign -i split-0.tx -o signout-0.tx
${gcmd} clerk sign --program ./clawback-escrow.teal -i split-1.tx -o signout-1.tx
${gcmd} clerk sign -i split-2.tx -o signout-2.tx
${sandbox} copyFrom signout-0.tx
${sandbox} copyFrom signout-1.tx
${sandbox} copyFrom signout-2.tx
echo "cat signout-0.tx signout-1.tx signout-2.tx  > signout.tx"
cat signout-0.tx signout-1.tx signout-2.tx  > signout.tx
${sandbox} copyTo signout.tx
echo "${gcmd} clerk rawsend -f signout.tx"
${gcmd} clerk rawsend -f signout.tx

rm *.tx
```

APPENDIX D: Extract of permission smart contract validations

```
gtxn 1 Sender
gtxn 2 Receiver
==
// verify the fee amount is good
gtxn 2 Amount
gtxn 1 Fee
>=
&&
bz failed

//Not allowed to transfer more than 50 units
gtxn 1 AssetAmount
int 50
<=
bz failed
//now check if sender is in the whitelist
int 0 // sender
gtxn 0 ApplicationID
byte "whitelisted" // load the local whitelist
app_local_get_ex
//==   -> Do not compare. We are checking that the account has a "whitelisted" attribute with value int:1
bz failed // there must be a global whitelist of accounts to validate the receiver

//gtxn 2 Receiver
//now check if asset receiver is in the whtielist
int 1 // asset receiver account
gtxn 0 ApplicationID
byte "whitelisted" // load the global whitelist
app_local_get_ex
//==
bz failed // there must be a global whitelist of accounts to validate the rece
```

APPENDIX E: Transfer from non-whitelisted account test bash script

```bash
#!/bin/bash

date '+keyreg-teal-test start %Y%m%d_%H%M%S'

set -e
set -x
set -o pipefail
export SHELLOPTS

# Directory of this bash program
DIR="$( cd "$( dirname "${BASH_SOURCE[0]}" )" >/dev/null 2>&1 && pwd )"

#gcmd="../../goal -d ../test/Primary"
#gcmd2="../../goal -d ../test/Node"

sandbox="./sandbox"
gcmd="./sandbox goal"
gcmd2="./sandbox goal -d ../Primary"

# Get one account from each node
#ACCOUNT=$(${gcmd} account list|awk '{ print $2 }'|head -n 1)
ACCOUNT=YTJH3JGHZ2BCKJNXUMXR2QQR6BFKX7T7FSGAQRUSKKY7TCA2RDQCUFXIKM
ACCOUNT2=$(${gcmd2} account list|awk '{ print $2 }'|head -n 1)
ESCROW=$(${gcmd} clerk compile clawback-escrow.teal | awk '{ print $2 }'|tail -n 1)
ESCROW="${ESCROW//$'\r'/}"
ASSET_AMOUNT=$((RANDOM % 51))

echo "Account1="$ACCOUNT
echo "ACCOUNT2="$ACCOUNT2
echo "ESCROW="$ESCROW
echo "ASSET_AMOUNT="$ASSET_AMOUNT

${gcmd} app call --app-id 32  --app-arg "str:transfer" --app-account ${ACCOUNT2} --from $ACCOUNT --out=unsginedtransaction1.tx  # smart contract call
${gcmd} asset send -a $ASSET_AMOUNT --assetid 30 -f ${ACCOUNT} -t ${ACCOUNT2} --clawback ${ESCROW}  --out=unsginedtransaction2.tx  # transfer from account 1 to account 2 using clawback account (ESCROW)
${gcmd} clerk send --from=$ACCOUNT --to=${ESCROW} --amount=1000 --out=unsginedtransaction3.tx #

#make output transactions accessible outside sandbox
${sandbox} copyFrom unsginedtransaction1.tx
${sandbox} copyFrom unsginedtransaction2.tx
${sandbox} copyFrom unsginedtransaction3.tx

cat unsginedtransaction1.tx unsginedtransaction2.tx unsginedtransaction3.tx > combinedtransactions.tx
${sandbox} copyTo combinedtransactions.tx
${gcmd} clerk group -i combinedtransactions.tx -o groupedtransactions.tx
${sandbox} copyFrom groupedtransactions.tx
${gcmd} clerk split -i groupedtransactions.tx -o split.tx
${gcmd} clerk sign -i split-0.tx -o signout-0.tx
${gcmd} clerk sign --program ./clawback-escrow.teal -i split-1.tx -o signout-1.tx
${gcmd} clerk sign -i split-2.tx -o signout-2.tx
${sandbox} copyFrom signout-0.tx
${sandbox} copyFrom signout-1.tx
${sandbox} copyFrom signout-2.tx
echo "cat signout-0.tx signout-1.tx signout-2.tx  > signout.tx"
cat signout-0.tx signout-1.tx signout-2.tx  > signout.tx
${sandbox} copyTo signout.tx
echo "${gcmd} clerk rawsend -f signout.tx"
${gcmd} clerk rawsend -f signout.tx

rm *.tx

#write output to file
#/4-transfer.sh > $(date +"%Y_%m_%d_%I_%M_%p")debug.log
```

APPENDIX F: Transfer to non-whitelisted account test bash script

```bash
#!/bin/bash

date '+keyreg-teal-test start %Y%m%d_%H%M%S'

set -e
set -x
set -o pipefail
export SHELLOPTS

# Directory of this bash program
DIR="$( cd "$( dirname "${BASH_SOURCE[0]}" )" >/dev/null 2>&1 && pwd )"

sandbox="./sandbox"
gcmd="./sandbox goal"
gcmd2="./sandbox goal -d ../Primary"

# Get one account from each node
ACCOUNT=$(${gcmd} account list|awk '{ print $2 }'|head -2| tail -1)
ACCOUNT2=YTJH3JGHZ2BCKJNXUMXR2QQR6BFKX7T7FSGAQRUSKKY7TCA2RDQCUFXIKM

ESCROW=$(${gcmd} clerk compile clawback-escrow.teal | awk '{ print $2 }'|tail -n 1)
ESCROW="${ESCROW//$'\r'/}"
ASSET_AMOUNT=$((RANDOM % 51))

echo "Account1="$ACCOUNT
echo "ACCOUNT2="$ACCOUNT2
echo "ESCROW="$ESCROW
echo "ASSET_AMOUNT="$ASSET_AMOUNT

${gcmd} app call --app-id 32  --app-arg "str:transfer" --app-account ${ACCOUNT2} --from $ACCOUNT --out=unsginedtransaction1.tx  # smart contract call
${gcmd} asset send -a $ASSET_AMOUNT --assetid 30 -f ${ACCOUNT} -t ${ACCOUNT2} --clawback ${ESCROW}  --out=unsginedtransaction2.tx  # transfer from account 1 to account 2 using clawback account (ESCROW)
${gcmd} clerk send --from=$ACCOUNT --to=${ESCROW} --amount=1000 --out=unsginedtransaction3.tx # Pay the ESCROW account for the transfer transaction

#make output transactions accessible outside sandbox
${sandbox} copyFrom unsginedtransaction1.tx
${sandbox} copyFrom unsginedtransaction2.tx
${sandbox} copyFrom unsginedtransaction3.tx

cat unsginedtransaction1.tx unsginedtransaction2.tx unsginedtransaction3.tx > combinedtransactions.tx
${sandbox} copyTo combinedtransactions.tx
${gcmd} clerk group -i combinedtransactions.tx -o groupedtransactions.tx
${sandbox} copyFrom groupedtransactions.tx
${gcmd} clerk split -i groupedtransactions.tx -o split.tx
${gcmd} clerk sign -i split-0.tx -o signout-0.tx
${gcmd} clerk sign --program ./clawback-escrow.teal -i split-1.tx -o signout-1.tx
${gcmd} clerk sign -i split-2.tx -o signout-2.tx
${sandbox} copyFrom signout-0.tx
${sandbox} copyFrom signout-1.tx
${sandbox} copyFrom signout-2.tx
echo "cat signout-0.tx signout-1.tx signout-2.tx  > signout.tx"
cat signout-0.tx signout-1.tx signout-2.tx  > signout.tx
${sandbox} copyTo signout.tx
echo "${gcmd} clerk rawsend -f signout.tx"
${gcmd} clerk rawsend -f signout.tx

rm *.tx
echo "Removing ACCOUNT=""$ACCOUNT"
OUTPUT=$(${gcmd} account delete -a "$ACCOUNT2")
echo "RESULT=""$OUTPUT"

```

APPENDIX G: Transfer outside of allowed asset amount range test bash script

```
#!/bin/bash

date '+keyreg-teal-test start %Y%m%d_%H%M%S'

set -e
set -x
set -o pipefail
export SHELLOPTS

# Directory of this bash program
DIR="$( cd "$( dirname "${BASH_SOURCE[0]}" )" >/dev/null 2>&1 && pwd )"

sandbox="./sandbox"
gcmd="./sandbox goal"
gcmd2="./sandbox goal -d ../Primary"

# Get one account from each node
ACCOUNT=$(${gcmd} account list|awk '{ print $2 }'|head -n 1)
ACCOUNT2=$(${gcmd2} account list|awk '{ print $2 }'|head -n 1)
ESCROW=$(${gcmd} clerk compile clawback-escrow.teal | awk '{ print $2 }'|tail -n 1)
ESCROW="${ESCROW//$'\r'/}"
ASSET_AMOUNT=$((RANDOM % 151 - 50))

echo "Account1="$ACCOUNT
echo "ACCOUNT2="$ACCOUNT2
echo "ESCROW="$ESCROW
echo "ASSET_AMOUNT="$ASSET_AMOUNT

${gcmd} app call --app-id 32  --app-arg "str:transfer" --app-account ${ACCOUNT2} --from $ACCOUNT --out=unsginedtransaction1.tx  # smart contract call
${gcmd} asset send -a $ASSET_AMOUNT --assetid 30 -f ${ACCOUNT} -t ${ACCOUNT2} --clawback ${ESCROW}  --out=unsginedtransaction2.tx  # transfer from account 1 to account 2 using clawback account (ESCROW)
${gcmd} clerk send --from=$ACCOUNT --to=${ESCROW} --amount=1000 --out=unsginedtransaction3.tx # Pay the ESCROW account for the transfer transaction

#make output transactions accessible outside sandbox
${sandbox} copyFrom unsginedtransaction1.tx
${sandbox} copyFrom unsginedtransaction2.tx
${sandbox} copyFrom unsginedtransaction3.tx

cat unsginedtransaction1.tx unsginedtransaction2.tx unsginedtransaction3.tx > combinedtransactions.tx
${sandbox} copyTo combinedtransactions.tx
${gcmd} clerk group -i combinedtransactions.tx -o groupedtransactions.tx
${sandbox} copyFrom groupedtransactions.tx
${gcmd} clerk split -i groupedtransactions.tx -o split.tx
${gcmd} clerk sign -i split-0.tx -o signout-0.tx
${gcmd} clerk sign --program ./clawback-escrow.teal -i split-1.tx -o signout-1.tx
${gcmd} clerk sign -i split-2.tx -o signout-2.tx
${sandbox} copyFrom signout-0.tx
${sandbox} copyFrom signout-1.tx
${sandbox} copyFrom signout-2.tx
echo "cat signout-0.tx signout-1.tx signout-2.tx  > signout.tx"
cat signout-0.tx signout-1.tx signout-2.tx  > signout.tx
${sandbox} copyTo signout.tx
echo "${gcmd} clerk rawsend -f signout.tx"

${gcmd} clerk rawsend -f signout.tx


rm *.tx
```

APPENDIX H: Transfer without payment to escrow test bash script

```
#!/bin/bash

date '+keyreg-teal-test start %Y%m%d_%H%M%S'

set -e
set -x
set -o pipefail
export SHELLOPTS

# Directory of this bash program
DIR="$( cd "$( dirname "${BASH_SOURCE[0]}" )" >/dev/null 2>&1 && pwd )"

#gcmd="../../goal -d ../test/Primary"
#gcmd2="../../goal -d ../test/Node"

sandbox="./sandbox"
gcmd="./sandbox goal"
gcmd2="./sandbox goal -d ../Primary"

# Get one account from each node
ACCOUNT=$(${gcmd} account list|awk '{ print $2 }'|head -n 1)
ACCOUNT2=$(${gcmd2} account list|awk '{ print $2 }'|head -n 1)
ESCROW=$(${gcmd} clerk compile clawback-escrow.teal | awk '{ print $2 }'|tail -n 1)
ESCROW="${ESCROW//$'\r'/}"
ASSET_AMOUNT=$((RANDOM % 51))

echo "Account1="$ACCOUNT
echo "ACCOUNT2="$ACCOUNT2
echo "ESCROW="$ESCROW
echo "ASSET_AMOUNT="$ASSET_AMOUNT

${gcmd} app call --app-id 32  --app-arg "str:transfer" --app-account ${ACCOUNT2} --from $ACCOUNT --out=unsginedtransaction1.tx  # smart contract call
${gcmd} asset send -a $ASSET_AMOUNT --assetid 30 -f ${ACCOUNT} -t ${ACCOUNT2} --clawback ${ESCROW}  --out=unsginedtransaction2.tx  # transfer from account 1 to account 2 using clawback account (ESCROW)
#${gcmd} clerk send --from=$ACCOUNT --to=${ESCROW} --amount=1000 --out=unsginedtransaction3.tx # Pay the ESCROW account for the transfer transaction


#make output transactions accessible outside sandbox
${sandbox} copyFrom unsginedtransaction1.tx
${sandbox} copyFrom unsginedtransaction2.tx
${sandbox} copyFrom unsginedtransaction3.tx

cat unsginedtransaction1.tx unsginedtransaction2.tx unsginedtransaction3.tx > combinedtransactions.tx
${sandbox} copyTo combinedtransactions.tx
${gcmd} clerk group -i combinedtransactions.tx -o groupedtransactions.tx
${sandbox} copyFrom groupedtransactions.tx
${gcmd} clerk split -i groupedtransactions.tx -o split.tx
${gcmd} clerk sign -i split-0.tx -o signout-0.tx
${gcmd} clerk sign --program ./clawback-escrow.teal -i split-1.tx -o signout-1.tx
${gcmd} clerk sign -i split-2.tx -o signout-2.tx
${sandbox} copyFrom signout-0.tx
${sandbox} copyFrom signout-1.tx
${sandbox} copyFrom signout-2.tx
echo "cat signout-0.tx signout-1.tx signout-2.tx  > signout.tx"
cat signout-0.tx signout-1.tx signout-2.tx  > signout.tx
${sandbox} copyTo signout.tx
echo "${gcmd} clerk rawsend -f signout.tx"
${gcmd} clerk rawsend -f signout.tx

rm *.tx
```