TU WIEN Informatics

Smart Economy

A Blockchain Solution – Automation of Austrian tax system

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Wirtschaftsinformatik

eingereicht von

Ing. Oliver Steizinger, BSc.
Matrikelnummer 01127076

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Privatdoz. Mag.rer.soc.oec. Dipl.-Ing. Dr.techn. Weippl Edgar

Wien, 12. Dezember 2021

Oliver Steizinger	Weippl Edgar

TU WIEN Informatics

Smart Economy

A Blockchain Solution – Automation of Austrian tax system

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Business Informatics

by

Ing. Oliver Steizinger, BSc.
Registration Number 01127076

to the Faculty of Informatics

at the TU Wien

Advisor: Privatdoz. Mag.rer.soc.oec. Dipl.-Ing. Dr.techn. Weippl Edgar

Vienna, 12th December, 2021

Oliver Steizinger	Weippl Edgar

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Ing. Oliver Steizinger, BSc.

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 12. Dezember 2021

Oliver Steizinger

Abstract

Blockchain and smart contracts are rising technologies in modern computer science. This enables a new type of application where financial products and services can be implemented as executable code. Accounting is an activity with still a lot of manual work which is necessary to calculate profit and further tax values. Bills are forwarded to the accountant who fed complex accounting software with the bill data and report back the payable tax amount. This leads us to the research question of this thesis, creating a decentralized application to automate tax systems like in Austria to demonstrate the possibilities of such systems. The system automates the manual processes of tax accounting and the transaction of the tax amount to the tax office. The thesis starts with theoretical research about blockchain systems and economic models, followed by the creation of a simple model of economic interactions necessary for tax calculations. Subsequently, the smart contract is implemented with the Solidity smart contract programming language on the Ethereum blockchain. Furthermore, this contracted is implemented on a web-based decentralized app to grant users access to the system.[1] The smart contract is published on Etherscan and the decentralized app is accessible via MetaMask on kovan network. [2] Metamask is a browser extension to connect an Ethereum wallet to a website based app and interact with smart contract functions. [met]

The thesis demonstrates how tax law can be implemented within a smart contract programming language. It includes implementations of income tax, VAT, social insurance, insurance tax, corporate tax, wage tax and local tax according to Austrian tax law.

The implementation demonstrates how an automated tax system could work, but it also shows the weakness of the current Ethereum live chain. With an emulation of transactions, the capabilities of the system are tested. The results show that more development is necessary regardless of the scalability and transaction costs of blockchains, especially the Ethereum blockchain. With second-layer solutions and ETH2, the outlook seems promising for further improvement of decentralized systems.

[1] http://www.smarteconomy.at

[2] https://kovan.etherscan.io/address/0x52B0a2531Bd462b67D9c959AC0Ac7c17f2851717#code

Contents

CHAPTER 1

Introduction

1.1 Motivation and problem statement

Business informatics at TU-Wien is a study of informatics, economics and the connection between these disciplines. From the perspective of a computer scientist, the models used to calculate and predict the economy seem simple in the first place but can evolve to complex structures and they are always influenced by social phenomena. With the blockchain technology and smart contracts, these two disciplines can be directly connected. In November 2015, Peters and Panayi discussed how Blockchain technologies and smart contracts could be the future of transaction processing [PP15]. Ali, Ally, Clutterbuck and Dwivedi released a literature review about the state of blockchain technology in the financial service sector five years later in October 2020 [AACD20]. It is interesting to see how scientific research is evolving and how much research papers are created in such short timeframe. This brings us to the motivation for this master thesis: We want to create a small economic system with hundred participants within the decentralized Ethereum blockchain platform and build a test environment to find out the limitations in terms of automatability and performance of an Ethereum smart contract by testing it on a local EVM and Truffle. This simulation intends to demonstrate how an automated economic system could work. It will not be designed to predict future economic events. The actual economic system is not entirely automated and transparent, and therefore we will create a system that meets these requirements and can thus be used for simulations. Cryptocurrencies with smart contract systems are able to create automated cash flows of business processes which need third person entities in our current economic system. With a decentralized system it is possible to eliminate the middleman in the economy processes that would lead us to a “trustless” economy system with automated interactions between multiple stakeholders. In this thesis we want to focus on the topic, how a government could create a smart contract on a permissionless blockchain, to collect their taxes automatically. Permissionless blockchain means that neither the functionality nor

the consensus algorithm is controlled by a central entity like defined in the paper of Peters and Panayi [PP15].

Ethereum and its smart contract system are a good example of this possibility. Hartel, Homoliak and Reijsbergen analyzed existing smart contracts in their paper "An Empirical Study into the Success of Listed Smart Contracts in Ethereum" [PP19]. Ethereum has already implemented some useful decentralized finance applications like the DAI stable coin. DAI is an Ethereum token that holds up the price of 1$ relatively strongly and is not affected by price fluctuations. The price was always between 0.96$ and 1.093$ since creation of the token and stabilized even more in 2021. As well there are projects with governance tokens which are used to give the owner of the token a right to influence decisions of further development. With all these tools it should be possible to program a simple economic model within the smart contract system. For this "smart economy" system, a test environment has to be implemented that is used to simulate economic interactions between stakeholders.

One of the biggest problems of permissionless blockchain systems like Ethereum right now is the scalability in terms transactions. Therefore, the simulation will focus on the ability of processing transactions with the current technology stack of Ethereum and how the system behaves with interactions on a larger scale. That means we will automatically create transactions to find out the limitations of a local EVM and relate these findings to the performance of the live-chain. Because of the high transaction costs, we will not test the system on the main network of Ethereum.

The implementation of the smart contract and a user interface leads to an automated tax system which is able to replace conventional bookkeeping systems. In theory a government could create a local blockchain for their residents and user a similar smart contract to collect taxes automatically. In our case, the government would have to force everybody to use this method because there are no interfaces to traditional tax systems. Realistically such systems need to fit into the existing system to evolve slowly.

We will examine the functionality of the program code with tests to see if the tax is redirected correctly to the stakeholder of the system. Besides the automatic transaction we calculate taxes traditionally and compare the results. The taxes are calculated differently due to the nature of a transaction. For example, the smart contract will provide a function for business addresses to pay wage to an employee. In this case, all wage related taxes are calculated and transferred directly to the tax office. In case of buying and selling products, a business address is able to create digital bills which then can be paid by customers. The VAT is transferred depending on the status of the customer (business or private).

Furthermore, we use this code to analyze how the current state of the Ethereum chain would react to a larger scale system like a tax system.

Furthermore, we use this code to analyze how the current state of the Ethereum chain would react to a larger scale system like a tax system.

1.2 Aim of the work

This thesis will start with a theoretical introduction that focuses on the blockchain and Ethereum smart contracts. We want to show how Ethereum and comparable systems, could be used to improve and automate our economic system. The main part of the master thesis consists of three practical tasks which depend on each other. These tasks will be the milestones of the research and will be finished one after another.

The first task of the practical work is to create a simple meta model that represents the stakeholders of our economic system in Austria from the perspective of financial interactions between them. I start with a simpler model of a small town and evolve it to a general model for a larger context. This model is used to build a class diagram and furthermore it is used as model for the smart contract. A smart contract is a program with defined functions which is stored on an Ethereum blockchain in our case. Ayman, Roy, Alipour and Laszka released a paper about smart contract development from the perspective of developers on social media [ARAL20]. The paper shows the dominance of smart contract development on the Ethereum platform which is the main reason why we pick Ethereum as smart contract development tool. Participants of the network are able to access this functionality to interact with the program and transfer their taxes directly to the tax office. All participants of a national economy will be considered to show how a nationwide decentralized economy could work. It is essentially a model of the business cycle [wikb] in modern economies comprising households, companies, state and capital collection points like banks [wika]. Social insurance is added as part of the economy in Austria. Modeling international trade relationships like import or export of goods is not part of this research project but is considered in the theoretical part. Internal processes of companies are not part of the research.

The second task of this project will be the proof of concept implementation of the smart contract(s) on the Ethereum network [eth]. For this purpose, the program language Solidity is used, which is an object-oriented, high-level language for implementing smart contracts [sol]. The contracts are deployed and tested on a public Ethereum test network (Ropsten, KOVAN). Moreover, a WEB3-Page is created to make the contract functions accessible in the browser. Di Angelo and Salzer provide a paper with tools for analyzing Ethereum smart contracts which can be used to find a fitting tool for the smart contract programming part [AS19].

In the third and last task, we will test the system and simulate a sample economy. It will be necessary to create our own test environment and a private Ethereum test network. An algorithm to simulate business interactions has to be defined and implemented to the system.

Finally, the results of the tests will be analyzed and real world impact will be deduced. The goal of this thesis is a smart contract implementation for a government which is used to collect taxes. Participants of an economy must interact with this smart contract to meet the defined tax requirements without manual calculations. Basically it could be

used for automation of tax settlement and should reduce the work for tax offices. We are exploring the limits of a single smart contract used by a set of participants concurrently.

1.2.1 Simplifications

This implementation assumes that all transactions are operated within the Ethereum network and without hard cash. This is a major political barrier but it is not important for a feasibility study.

Capital collecting points, investment associations and other insurance companies than social insurance will not be considered for the implementation, for reasons of simplification.

Branch specific taxes which need product amounts as an external input to calculate the tax are not implemented but discussed in the thesis. To automate such taxes an oracle is necessary.

By reason that enterprises can exist with many different characteristics, the implementation is a simplification of the entire reality with the possibility to extend. Sole proprietorship, partnership (OG, KG), corporation (GmbH, AG) and other corporate forms are treated equally.

1.3 Methodological approach

The methodological approach comprises the following parts:

- **Systematic literature review**

  Basic information about the blockchain and Ethereum smart contracts has to be gathered to provide an overview of the topic. An economy model has to be created and its limitations discussed.

- **Economy model**

  The created economy model has to be converted to a class diagram.

- **Smart contract implementation**

  The implementation of the smart contracts will be performed with the Design Science methodology and it will produce the following two artifacts:

  - Solidity implementation (smart contracts)
  - Web3 browser integration (JavaScript code)

- **Testing and emulation**

  Five steps are necessary to test the created software artifacts with meaningful data:

  - Gathering data about business interactions in Austria
  - Creating test environment for smart contracts

  - Creating emulation algorithms with gathered data
  - Running emulation
  - Validation

- **Analysis**

  In the analysis, the results of the emulation are related to the real world, considering both technical and economic perspective. Based on the theoretical insights derived from the economic model, possible future trajectories of our economic system are reflected upon.

- **Further research**

  In the final part, the created contract and how it could be used in further research is outlined.

1.4 Literature review

The first method used in this thesis is a literature review of blockchain systems and economy models. This method is used to find all available research for related topics and choose which research papers can contribute to the thesis. The most important research contributions are selected and summarized in the theoretical part of the thesis. The following steps are based on the research methods lecture of Associate Prof. Dr.-Ing. Stefan Schulte, which can be summarized as planning, conducting the review and reporting. To cover specialties of systematic literature reviews within the domain of software engineering, the journal article "Lessons from applying the systematic literature review process within the software engineering domain" was reviewed. This article concludes that the basic steps of a systematic literature review appear as relevant to software engineering as they do to medicine, and it offers insight into which modifications improve the research process. [BKB+07]

- **Formulate your research question**

  This master thesis is a practical solution for an economic problem. We want to show and analyze how a government could use a smart contract for tax collecting purposes. Accordingly, two research topics have to be researched: first, a valid literature background for the economic model is necessary; and second, the technical solution is state of the art. As the understanding of the problem increases during the development of the research protocol, the research question will be revised afterwards.

- **Establish a pilot**

  A pilot study is used to build a review protocol. The pilot study helps to clarify the research questions and the data collected within the review protocol. Ideally, an external reviewer should validate the process.

- **Choose appropriate search keywords**

  With the definition of the software artifact extended by blockchain technology terms, the search keywords can be specified to find relevant research. After a search cycle, the keywords are reviewed and eventually extended. Because of the well-defined problem definition and the short history of blockchain systems a manageable amount of research papers is expected.

- **Conduct the search and collect studies**

  After an overview of the search results of different sources, the abstracts of relevant research papers are analyzed. It is important to search with multiple sources to obtain all primary studies of the topics.

- **Select relevant studies**

  The most valuable research is selected to be included in this master thesis. Because of poor standards of software engineering abstracts the conclusion will be added to the review process. In addition to traditional literature, we use grey literature and community inputs for the development of the smart contract.

- **Analyze primary studies**

  Selected studies are analyzed to support the validity of the software artifact.

- **Report on the results**

  The entire research process is consolidated in a report to give the reader the possibility to validate the research process.

The process of the systematic literature review is documented with five key elements. The review protocol specifies the research question and the methods used. A search strategy is defined to find all related research. To assess the rigor and completeness of the systematic literature review, documentation is needed. Explicit criteria with explicit inclusions and exclusion criteria are required. For the analysis of the reviewed papers, an information specification is created, specifying the information to be obtained from each study including criteria to evaluate (Attachment 1 - Review protocol).

1.5 Economy model

With the findings of the theoretical research we create an economic model including the most relevant economic participants and their financial relationships. A simple business example is used to monitor errors within the model by testing all possible transactions between all economic stakeholders. The model has to include all participants which are part of the defined tax transactions, all values stored for further tax calculations and the transactions between the participants. It is correct when all from the government defined tax calculations can be calculated within the model. The goal is an entity-relationship-model which can be used for the implementation of the smart contract.

1.6 Smart contract implementation

For the smart contract and the web3 page implementation, the Design Science method is used. Based on the Design Science IS Research Framework (Figure 1.1 - Design Science IS Research Framework), Hevner et al. (2004) created seven guidelines to create an information system artifact. [HMPR04]

- **Guideline 1: Design as an Artifact**

  "Design-science research must produce a viable artifact in the form of a construct, a model, a method, or an instantiation."

  For the given problem statement, the product is a smart contract on the Ethereum blockchain that fulfills the requirements of a software artifact.

- **Guideline 2: Problem Relevance**

  "The objective of design-science research is to develop technology-based solutions to important and relevant business problems."

  The problem relevance is discussed in the research part of the thesis. Essentially, the artifact can be used for automation of tax settlement and should reduce the work for tax offices.

- **Guideline 3: Design Evaluation**

  "The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods."

  Due to the nature of the Ethereum blockchain where computing power is costly, the evaluation of the smart contract is a main aspect of the developing process. Accordingly, the source code needs to be as simple as possible, with a low number of data transfers. The code will be reviewed as an iterative process until no major improvements can be identified.

- **Guideline 4: Research Contributions**

  "Effective design-science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations, and/or design methodologies."

  This master thesis is a feasibility study of an automated economy system based on current decentralized blockchain technologies.

- **Guideline 5: Research Rigor**

  "Design-science research relies upon the application of rigorous methods in both, the construction and evaluation of the design artifact."

  A final systematic approach will be defined with the created economy model. Essentially, a construct with all economic participants is created and the business functions are added consecutively.

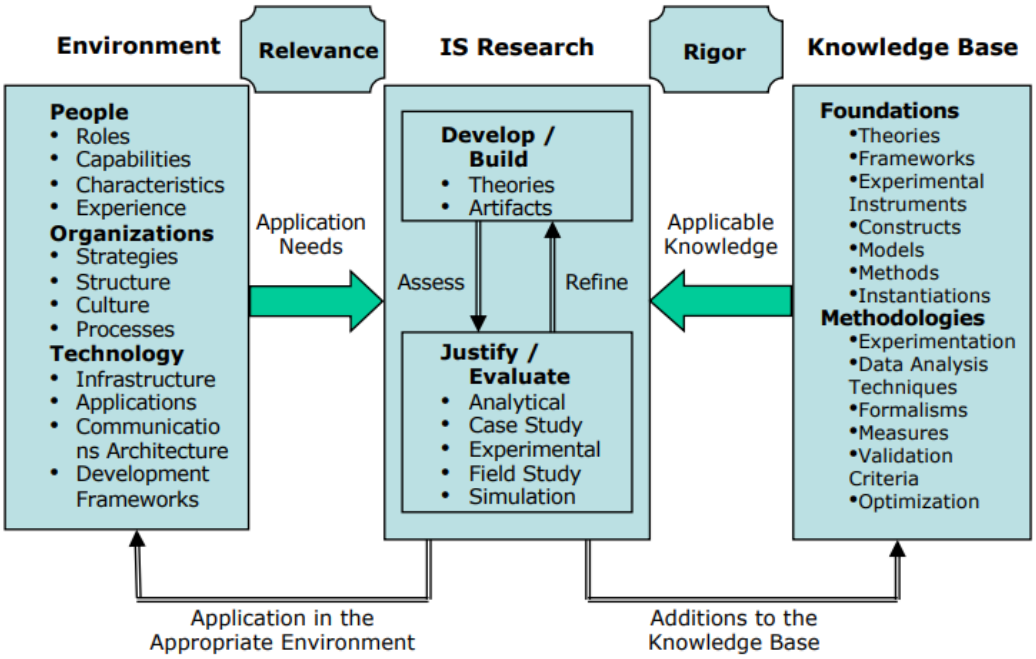


Figure 1.1: Design Science IS Research Framework

- **Guideline 6: Design as a Search Process**

  "The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment."

- **Guideline 7: Communication of Research**

  "Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences."

  The Ethereum blockchain explorer Etherscan provides a service to give people the possibility to validate the code of a smart contract. Without this validation, nobody would trust the smart contract because backdoors would be possible. The code is publically available on Etherscan and GitLab.

1.7 Testing and emulation

The created software artifact is tested with an emulation of the real-world problem. This iterative process is performed until the result is proven satisfactory by reaching key indicators. The emulation will be performed on a local test environment and the results will be adopted to the live Ethereum chain. The purpose of the emulation in my thesis is to test the performance of the system and find scalability issues. Furthermore, the emulation should prove the concept of the created software artifact and make it closer to

reality, therefore being easier to understand for people by being imaginable how such systems could work. To compare the results with the real world, public data about the number of transactions carried out in Austria is used.

CHAPTER 2

Blockchain – State of the art

2.1 Blockchain technology

Blockchain technology is an emerging field in the information industry, although most projects are still developing. The advantages of a decentralized ledger all over the world are interesting because it provides a secured, trusted and autonomous ecosystem for many purposes. Yuan and Wang (2018) discuss existing and potential ecosystems of Bitcoin and other cryptocurrencies in their article "Blockchain and Cryptocurrencies: Model, Techniques, and Applications". [YW18] The most government solutions in Blockchain literature are about Blockchain-Based voting systems. Yu, et al. introduced a "Platform-independent Secure Blockchain-Based Voting System" [YLS+18] and Spadafora, Longo and Sala proposed a "A Coercion-Resistant Blockchain-Based E-Voting Protocol with Receipts" [SLS20]. Other possible use cases for government services are discussed in the paper "Blockchain in the Government Technology Fabric" [Anw19]. Many scientific contributions are basic research and not focused on a specific problem. Lu, Xu, Bandara, Chen and Zhu introduced "Patterns for Blockchain-Based Payment Applications" as an overview of possible payment patterns and their challenges [LXB+21].

Technically speaking, a public blockchain is an authenticated data structure which can be read by everyone with an internet connection and where everyone can insert data for a fee. Interactions with the Blockchain are verified by the nodes and will be replicated to every single node all over the world. If somebody wants to read the data, it´s enough to read from one available connected node, because it´s a copy of the complete database. The system has to verify if the node is not on a stale branch or lagging behind. Writing is expensive because every node has to maintain the entire database. The biggest threat to the common Blockchain systems systems based on Nakamoto consensus, are so called 51% attacks. If one transaction validator has the majority of voting power, he can basically decide on his own what is happening with transactions on this Blockchain by censoring transactions or revert states which makes it vital to keep the transaction verification

as decentralized as possible. This leads us to the consensus algorithms in the current state of development. This will be explained with Ethereum as example even there exists more blockchains with similar consensus algorithms. Ethereum seems to be the project with the most ongoing development and fits well for this master thesis because it is well documented, has a broad community and many ongoing projects. Although this is not a complete list of decentralized consensus algorithms, nonetheless we will stick with the currently most common decentralized ones. Further research about consensus algorithms can be found in the paper "Blocks and Chains: Introduction to Bitcoin, Cryptocurrencies, and Their Consensus Mechanisms." [JSK+17] or more current in "Blockchain Consensus Algorithms: A Survey" [FMHC20].

2.1.1 Proof-of-work

Proof-of-work is the system used by the leading cryptocurrencies (Bitcoin, Ethereum) at present. The validation process is secured by computing power of the validation nodes. The process to add new blocks to the chain is called "Mining". Modern computers can be used to participate in the mining process by maintaining a node of the network and start mining Ethereum blocks. Each node in the Ethereum network collects new transaction requests in their local mempool that have not yet been committed in a block. Then it aggregates transactions depending on their size and propose a new block, a new system state of the chain to collect the transaction fees including. The node has to verify the validity of each transaction request and executes the code on the local copy of the EVM. Then the process of producing the Proof-of-Work "certificate of legitimacy" for the potential block starts were other nodes hear about the new block, verify the certificate and execute the transactions on their own EVM. All transactions of the new block are removed from the local mempool of all nodes. Because of the constantly used computing power to maintain the blockchain (mining) even there are no transactions, the energy consumption of PoW (Proof-of-Work) is very high [min]. Ethereums transaction costs are called gas fees. Yang, Murray, Rimba and Parampalli analyzed Ethereums gas mechanism in their paper "Empirically Analyzing Ethereum's Gas Mechanism" [YMRP19].

2.1.2 Proof-of-stake

Proof-of-stake is considered as the solution for energy problems of PoW blockchains. This consensus algorithm does not depend on computing power and therefore has a much better power efficiency. The node operators can partake in consensus by locking some Ether for a period of time instead of using computing power for mining constantly. The node still needs to be online permanent but small computers with low electric power consumption can be used for that. With PoS (Proof of Stake) rewards doesn't depend on computing power, the probability of being selected as a consensus leader to propose the next block is proportional to the stake relative to the total stake of the system. To avoid 51% attacks, the coins has to be distributed well over multiple staking node operators.

As previously mentioned, the leading cryptocurrencies are currently working with PoW,

although developers are working on solutions to increase efficiency, scalability and transaction numbers. For example, Ethereum is developing a PoS algorithm and plans to change the current Ethereum network to use PoS instead of PoW in the near future. With EIP-3675, the specification for the Ethereum improvement proposal to upgrade the consensus algorithm to proof-of-stake is already available on github github [eip]. In 2017, Poon and Buterin proposed the Plasma framework which is scalable to a significant amount of state updates per second (potentially billions) [PB17]. Another approach are sidechain mechanisms like zkRelay presented by Westerkamp and Eberhardt [WE20]. Robinson discusses the pros and cons of using the public Ethereum blockchain as a coordination chain for private sidechains in his paper [Rob19].

There are no governmental economic solutions in the cryptocurrency community at present given the early stage of development. In theory, a government could use a local blockchain to simplify some processes with their citizens. Soelman analyzed permissioned blockchains very detailed in his work “Permissioned Blockchains: A Comparative Study” [Soe21]. For example, blockchain systems could be used for voting or to trace origin of products. To get there, Governments all over the world need to catch up and regulate the crypto space to build up trust and make blockchains useable for average consumers. Without official certificates to prove that blockchains and smart contracts are safe, it is difficult to trust the systems as a non-expert. Without the trust of governments, decentralized economic systems cannot be realized under the current political circumstances.

This project provides a technical prove of such economic systems and offers an insight into how it could perform. The simulated system helps politicians and citizens to understand that such systems are already realizable, which makes it a social meaningful project to convince people of decentralized monetary systems.

2.2 How can Blockchain help us to automate economic processes?

Today, digitalization is at a point where the majority of businesses are participating. Therefore, we can expand our view of automation to a financial perspective. From the financial view, economic processes comprise a chain of transactions with two major issues, namely security and trust. At present, third parties take care of these issues, mostly banks or similar institutions. These third parties can be eliminated with a trustless blockchain system like Ethereum. Transactions are directly connected to real-world events and will be executed automatically. For example, a company could track the delivery of goods with the help of a smart contract system, deposit the payment to the smart contract and ensure that the payment will only be sent after the delivery is successful; otherwise, the payment could be sent back after a specified amount of time. Nevertheless an oracle is necessary to assess if the delivery was successful. Oracles feed smart contracts with external information[ora]. Abdeljalil Beniiche studied and described the widely used blockchain oracles and elaborated their potential role in "A Study of Blockchain Oracles" [Ben20].

Furthermore, from a financial view, the tax system in Austria is also a simple chain of transactions. This leads us back to the motivation of this master thesis: with the model of the tax system, a smart contract will be created and its limitations will be discussed. VAT in Austria is a good example of how a smart contract could be used for automation. If a company buys a product including VAT, the company can get the money back from the tax office because VAT only has to be paid by customers. This means that the first company pays the VAT to the second company, the second company pays the VAT to the tax office and the tax office pays the VAT back to the first company. A smart contract could reduce these three transactions to one smart transaction.

Due to the well-connected community and the larger number of research contributions, Ethereum smart contracts will be used for this master thesis.

2.3 Ethereum smart contracts

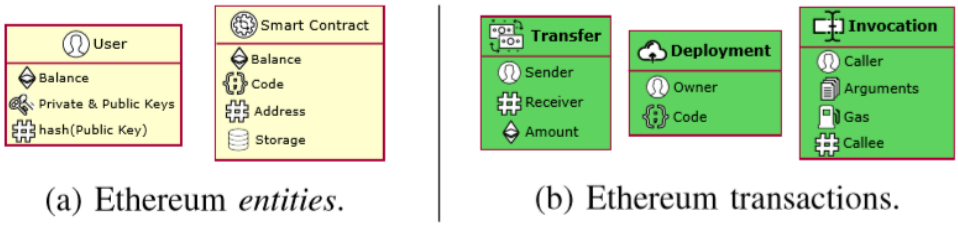
As a result of the literature research, studies about the Ethereum blockchain have been selected and reviewed. The paper "Blockchain for Trustworthy Coordination: A First Study with LINDA and Ethereum" stood out with its well-structured description of Ethereum. This chapter references section two of the mentioned research paper. [CMO18]

The Ethereum blockchain comprises a peer-to-peer network of nodes enacting a consensus protocol that lets them globally behave as a single-state machine. For execution of transactions, miners are responsible, which commits the results to blocks of data linked by hash chains, following the principles of the algorithm. The transaction information is stored in the blockchain on every running node, whereby the entire transaction history of the Ethereum platform can be viewed. Since Ethereum is a smart contract platform, transactions can contain more complex information than simply a value. These types of transactions are controlled by pre-defined smart contracts, which are described in a subsection of this chapter.

For the further part of this thesis, the abstractions used for Ethereum development are used when speaking about blockchain.

2.3.1 Entities & accounts

Ethereum has two types of entities that are stored in a map associating entity identifiers (addresses) to accounts (data). Entities can be end users or smart contracts. In both cases, entities comprise a data structure containing a balance and a secured storage area. The balance is the amount of ether, the Ethereum crypto currency owned by the account. The secured storage area contains arbitrary user data of the account. Smart contracts additionally expose a field containing the source code of the smart contract. The entities are depicted in "Figure 2.1 - Ethereum components" on the left side of the image.



(a) Ethereum *entities*. (b) Ethereum transactions.

Figure 2.1: Ethereum components

2.3.2 Transactions

There are three different types of transactions to be published by the user, which triggers a gossiping algorithm to spread the information to all participants of the consensus protocol. Depending on the specified gas fee, the transaction will be performed and validated by the consensus protocol participants within some seconds or minutes. This specially depends on the number of active transactions on the system. Transfers are simple transactions were an amount of ether is sent from one account to another. Such transaction comprises a sender address, a receiver address and the amount of ether sent. Deployment is a transaction where a user can upload a smart contract to the network to make it accessible for all participating accounts of the system. It includes the owner of the smart contract and the source code. The cost of such transactions depends on the length of the source code, because more storage space means higher transaction fees. Finally, an invocation of a smart contract is the most flexible sort of transactions because the details are defined in a smart contract. If a smart contract is deployed, every user is able to execute the public methods of a smart contract. Besides the sending and receiving account, the gas fee and the arguments defined in the smart contract are the parts of invocation transactions.

2.3.3 Smart contracts

Smart contracts are processes executing decentralized computations on the blockchain with the following properties:

- **Stateful:** each smart contract encapsulates its own state.
- **User-defined:** any user may publish a smart contract.
- **Reactive:** only users may trigger a smart contract.
- **Immutable:** smart contracts code cannot change.
- **Trustable:** no entity can tamper with the specification of a published smart contract, no user can lie about its smart contract invocations and the side effects caused by computations are guaranteed to produce a consistent change of the system state.

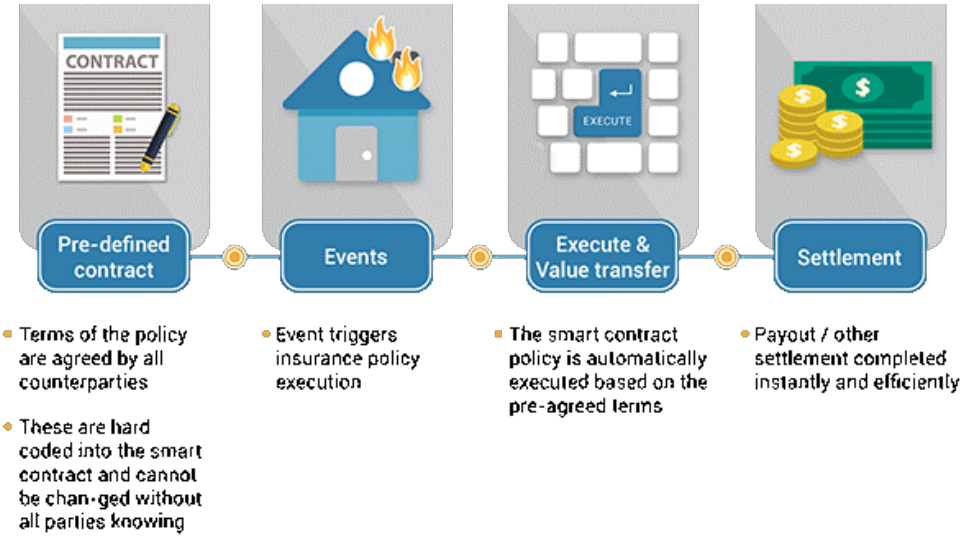


Figure 2.2: Smart contract example

- **Deterministic:** each computation always provides the same outputs if given same inputs, regardless of the actual execution node.
- **Decentralized:** there is no single, centralized coordinator governing the distributed execution of smart contracts, which happens concurrently.

Smart contracts are programmed with a quasi-Turing-complete language and they are virtually capable of implementing any computation. Smart contracts are objects in the OOP sense that interact with synchronous method calls. In “Figure 2.2 - Smart contract example”, a simple smart contract implementation with an event is shown.

2.3.4 Consensus, miners, and blocks

The consensus protocol makes every node in the blockchain participate in validation and consistency checks of transactions. It makes an arbitrary number of nodes perform exactly the same state transition for exactly the same state machine and prevents faulty nodes from creating inconsistencies.

Miners are the nodes participating on the consensus protocol. They are in charge of validation, consistency checking and including transactions in blocks. Agreement about the ordering of transactions is achieved by the consensus mechanism known as proof-of-work. Miners need to compete in solving a resource-intensive computational puzzle that grants the right to publish the new block containing the validated transactions, for which they are rewarded.

2.3.5 Miners & gas

The blockchain assumes that miners are rational agents and it promotes honesty by compensating their computational effort with the right to generate and claim crypto currency for each block successfully mined. Furthermore, miners are in charge of executing smart contracts and the deployment of messages and smart contracts. Because computational power is needed to store data on the blockchain, users have to pay to write into the decentralized database, which is lately reclaimed by the miners for executing the transaction. This means that users must endow transactions with a finite amount of gas (small amount of ether). The gas is spent while miners are executing the method (transaction) of the smart contract. If the computation fails due to a low amount of gas, all actions are reverted with no refund of the invested gas. Most user interfaces provide a useful calculation for the amount of gas to be used for the desired transaction.

2.3.6 Logs & API

Ethereum provides instructions to publish logs from smart contracts to represent the occurrence of some events to be stored within blocks to allow off-chain clients to inspect the blockchain. The Ethereum community has produced a number of high-level programming languages. One of the most common languages is Solidity, which has a JavaScript-like syntax and is similar to object-oriented programming. Solidity will be used for the smart contract implementation in this project.

2.4 Decentralized Applications (dApps)

DApps are applications interacting with a decentralized blockchain system. It is a smartphone or browser application with a blockchain as a database in the background. Events on the blockchain can trigger events in the application. This dApp technology makes blockchain accessible for the many because the front end can look like well-known applications. In this project, MetaMask is used to connect a browser to the blockchain platform and a web3 website with JavaScript is built to make the smart contract methods more easily accessible, as well as making the functionality of the system clearly visible.

2.5 Other smart contract systems

Besides Ethereum, there are many other smart contract systems with different approaches in development. At present, Ethereum seems to have the most community support and is on a good track to improve scalability issues, and it may change to a PoS concept to erase the problem with electric power consumption. If the reader is interested in gaining a deeper insight into the differences between smart contract systems implemented at present, Blockgeeks published a guide describing the functionalities of Ethereum, EOS, Stellar, Cardano, Neo and Hyperledger Fabric [Blo]. Although there are many more blockchain projects, this offers an insight into how multiple blockchain systems could exist

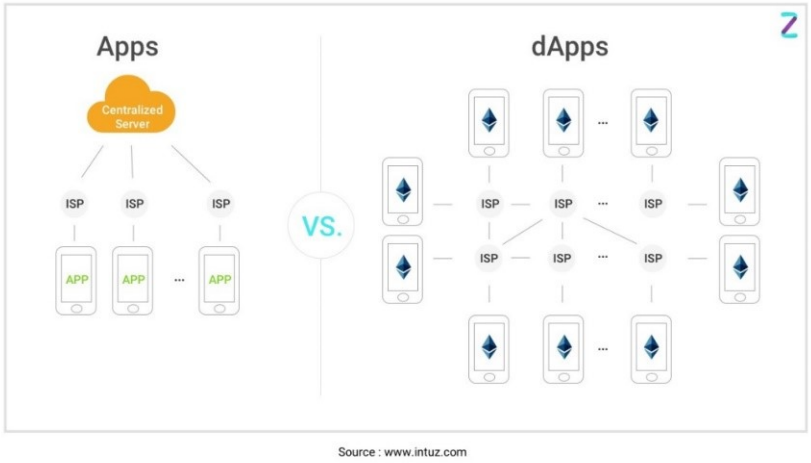


Figure 2.3: App and dApp comparison

for differing use cases. Because this master thesis only attempts to show the potential of smart contracts and is not implemented for eternity, the selection of the system is focused on the usability for programming tasks and community support.

CHAPTER 3

Business Cycle – Economy model in Austria

In this master thesis, a business cycle is considered as a model of all participants of the economy and the transactions between them. Furthermore, the research is restricted to a single country, namely Austria. Due to this restriction, the model will not be a closed cycle, which would have to include imports and exports of goods. Despite the limitations, money flows in a circular way between multiple types of entities; for example, companies, tax office, private customers and social insurance. The transactions between these entities are defined by the law and executed by the tax office. The aim of this chapter is to place these definitions into a business cycle model for further use in the practical part of the thesis.

In many other studies, the term business cycle has a different meaning, and thus to avoid confusion an explanation of what it means in other studies is provided. Many economic studies undertake analytical research about the economy on a large scale and over a long period of time [SLL17] [BI19]. Therefore, a business cycle is the development of economic measures over a period of time if you assume that the economy is a repeating process. In this master thesis, the development of economic measures is not relevant because it is focused on the practical solution of a process. The term business cycle is translated from the German word "Wirtschaftskreislauf" and refers to the circulation of money in economy. Therefore, this business cycle is a model of business participants and the transactions between them.

3.1 Participants of a business cycle

To build a fitting model for the purpose of this master thesis, a simple model of the business participants (Figure 3.1 - Business cycle [wika]) is used and will be extended

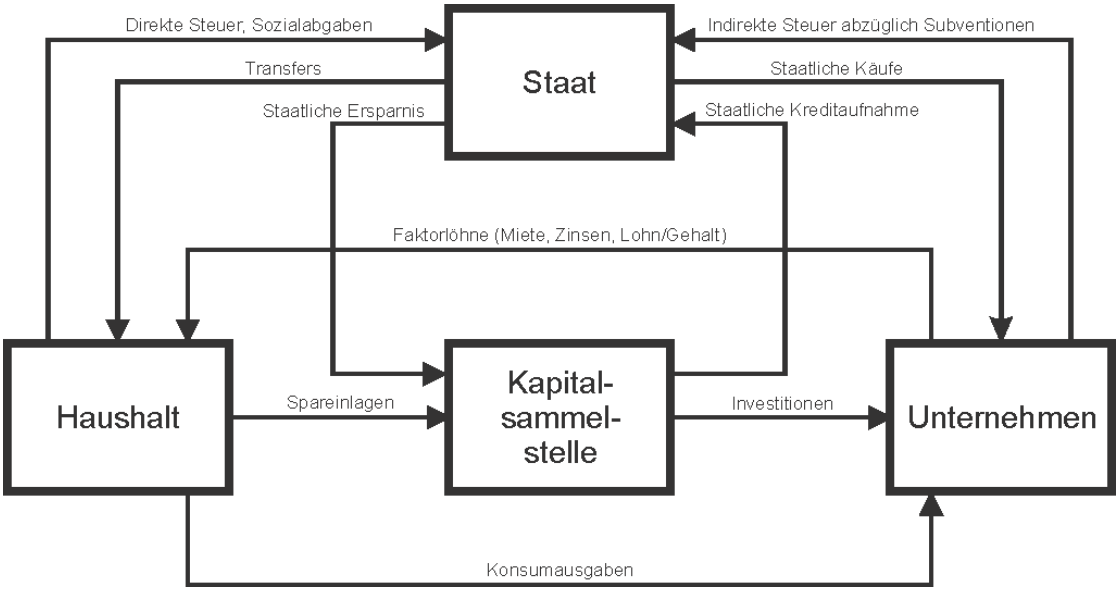


Figure 3.1: Business cycle [wika]

for the economy in Austria. It is a highly simplified model with no regards to different types of companies and with generalized transactions. Given that this master thesis has limited resources and aims to prove the concept rather than deliver a complete solution, a "full economy model" is not created; rather, it is more an abstraction of a business cycle. The main issue with the model in Figure 3.1 is the absence of social insurance. In Austria, a duty exists to have social insurance, which is an independent organization beside the government. Therefore, five types of business participants are identified.

3.1.1 Government

Besides social insurance, the government is the main tax collecting point in the economic environment. In Austria, the tax office collects several types of taxes from enterprises, capital collecting points and individual persons. These taxes are ruled by law and can change due to political decisions. In this section, an overview of tax types in Austria is provided. [Fin]

Income Tax - [German] Einkommensteuer (ESt)

Income tax is mandatory for every individual person with a main residence in Austria. The tax is based on the yearly income and higher income is taxed with higher income tax rates. The values in "Table 1 - Income tax rates" have been in effect since 2020. For each individual person, the tax rate is calculated for each income level. Accordingly, if an individual person earns 21,000€, the tax rate is separated into three levels. For the "first" 11,000€, no tax has to be paid, for the income above 11,000€ and below 18,000€ (i.e. a range of 7000€), 20% has to be paid (1,400€), and the remaining income part from 18,000€ to 21,000€ is taxed at 35%, whereby the tax for this income part is 1,050€.

Tax level income in Euro	Income Tax from 2020
11 000 and below	**0%**
above 11 000 to 18 000	**20%**
above 18 000 to 31 000	**35%**
above 31 000 to 60 000	**42%**
above 60 000 to 90 000	**48%**
above 90 000 to 1 000 000	**50%**
above 1 000 000	**55%**

Table 3.1: Income tax rates [Fin]

Accordingly, the sum of tax would be 2,450€ in this example, representing a calculated tax rate of 8.57% for an income of 21,000€.

Property Income Tax - [German] Immobilienertragsteuer (ImmoESt)

In Austria, one of seven income types is renting and leasing of property. Property income tax has some particular specifications that are not relevant for this research but is a separated tax in law. For example, the deduction of wear is especially regulated for property income tax. It regulates the operating life of an object regarding taxes. Property income tax is processed like regular income tax in case of an individual or corporate tax in case of an enterprise. Regarding technical feasibility, the actual percentages in the calculations do not really matter, but nevertheless we keep in mind that the project could be extended with multiple types of income tax calculations by copying the specific contract and change the hard-coded percentages. This would lead to a specific contracts for specific income types like property income, agriculture and forestry income, independent work income, commercial enterprise income and so on.

Capital Return Tax - [German] Kapitalertragsteuer (KESt)

The capital return tax is a tax for earnings from private capital assets. This tax is directly withheld from the capital collecting point (bank) and transferred to the tax office. The taxation is carried out depending on the type of income, with 25% for savings book and giro account interest or 27.5% for other investment income.

Corporate Tax - [German] Körperschaftsteuer (KÖSt)

Besides the income tax for individual persons, enterprises have to pay another kind of tax to the government. Enterprises are juristic persons of private law (for example AG, GmbH, Genossenschaften, Vereine) and juristic persons of public law (for example Gebietskörperschaften wie Bund, Länder und Gemeinden, Kammern, Sozialversicherungsträger, gesetzlich anerkannte Religionsgemeinschaften). Enterprises of public law only need to pay the tax if they operate a commercial business. Furthermore, there are exceptions for community services, culture supporting projects and similar activities. The rate of corporate tax is 25% of taxable income, and the percentage is independent of the total amount of income. Corporate tax is a linear tax rate in Austria. Unrestricted assessable

capital companies have a defined minimum tax. For easier understanding of the technical solution, this exemption will not be implemented in the first version of the system, like the property income tax exceptions. To simplify the programming amount of the first implementation, the number of similar tax calculations types is kept small.

Wage Tax - [German] Lohnsteuer (LSt)

In Austria, wage tax and income tax are essentially the same and they only differ by the agent who pays it in our model. Self-employed people have to pay income tax, in contrast to employees and retirees, who pay wage tax. Wage tax is withheld from the wage of the employee by the employers and directly paid to the tax office. Employees will only receive the net of tax value and do not have to pay the tax themselves. The tax rates are identical to income tax.

Value added tax (VAT) - [German] Umsatzsteuer (USt)

Corporations in Austria have to pay VAT for each product or service sold. Under normal conditions, the tax rate is 20% of the amount paid. In some exceptions, the rate is reduced to 13% (plants, arts, movies, sports, etc.) or 10% (renting, books, food, etc.). Corporations can reduce the amount of VAT by input tax reductions. Accordingly, if a company buys a product from another company and pays 20% VAT, they can reduce the amount of VAT paid to the tax office from their own sales. In other words, intermediate products that are used for the production of a product or service are free from VAT for companies when they use input tax reduction. The input tax reduction is calculated per quarter or month depending on the amount of sales. The VAT calculation has to be part of a bill.

Local Tax - [German] Kommunalsteuer (KommSt)

Local tax is collected from the local authorities within the legal basis of the Austrian government, similar to land tax. The tax rate is 3% of the defined basis assessment. All enterprises located in the area of local authorities are subject to local tax. Unlike land tax, local tax is not set by the local authorities themselves.

Insurance Tax - [German] Versicherungssteuer (VersSt)

Insurance tax has to be calculated by the insurer and paid to the tax office. The tax rate is based on the insurance type; for example, social insurance has a very low tax rate (1% to 2.5%), while property insurance has a higher rate (11%). In case of insurance, this master thesis will focus on social insurance corporations due to their special importance in Austria.

The following taxes are specialized on specific products and will not be part of the implementation. However, the implementation needs to provide a solution to enable implementations of specialized tax for multiple industries. The execution is like the local tax, where a percentage of a defined basis assessment is paid to the tax office but in this cases the defined basis assessment is an external value and would need an oracle to be fully automated. Another option is an interface for the user to input the corresponding

amounts. Because these kinds of taxes are similar in execution (i.e. paying a tax for an amount of product), only a brief explanation is provided.

Alcohol Tax - [German] Alkoholsteuer

Alcoholic goods produced in or imported to Austria are subject to alcohol tax.

Beer Tax - [German] Biersteuer

Similar to alcohol tax, all beer produced in or imported to Austria is subject to beer tax.

Digital Tax Law - [German] Digitalsteuergesetz 2020

Since January 2020, online advertising earnings from large advertising platforms are subject to digital tax.

Natural Gas Tax - [German] Erdgasabgabe

Natural gas used in Austria is subject to natural gas tax. The amount has to be calculated and reported to the tax office.

Flight Tax - [German] Flugabgabe (FlugAbgG)

Aircrafts operators have to pay flight tax for each passenger leaving an Austrian airport.

Land Acquisition Tax - [German] Grunderwerbsteuer (GrESt)

The acquisition of domestic estate is subject to land acquisition tax. The tax rate is essentially based on the money consideration, although in some cases it is based on the value of the estate.

Land Tax - [German] Grundsteuer (GrSt)

Land tax is set and collected yearly by the local authorities. Real estate is subject to land tax. To include this tax in our model, multiple types of government agents are necessary. Besides the central Austrian tax office – which collects most other taxes – multiple local authorities collect land tax with specific tax rates in a framework provided by the Austrian government.

Coal Tax - [German] Kohleabgabe

Like natural gas, coal used in Austria is subject to coal tax. The amount has to be calculated and reported to the tax office.

Motor Vehicle Tax - [German] Kraftfahrzeugsteuer (KfzSt)

Motor vehicles with a maximal acceptable weight over 3.5 tons are subject to motor vehicle tax. The tax rate is based on the maximal acceptable weight of the vehicle and it increases for heavier machines.

Petroleum Tax - [German] Mineralölsteuer (MÖSt)

Like natural gas and coal, petroleum used in Austria is subject to petroleum tax. The amount has to be calculated and reported to the tax office.

Norm Consumption Tax - [German] Normverbrauchsabgabe (NoVA)

Every vehicle that is licensed in Austria for the first time is subject to norm consumption tax.

Advertising Tax - [German] Werbeabgabe

Commercial advertising effort is subject of advertising tax. The tax rate is 5% of the defined basis assessment. It has to be paid by the advertiser, and if this is not possible because it is not located in Austria, the purchaser has to pay the tax. Furthermore, if the purchaser is not reachable, the one who benefits from the advertisement is responsible.

Summing up, different types of government agents are necessary to model the government participants. On the one hand, there is the Ministry of Finance, which collects tax carried out by the tax office. On the other hand, there are multiple local authorities (municipalities) guided by the laws introduced by the Ministry of Finance.

3.1.2 Social insurance

Although insurance is a much more complex topic and there are more insurance types available in Austria, this implementation will focus on social insurance in Austria. Social insurance is mandatory for each person with an income living in Austria and it is regulated by law, but is executed by independent social insurance institutions. In Austria, there are several social insurance institutions depending on the working condition of the person. For example, a self-employed person is insured with social insurance for the self-employed (SVS), whereas an employee is insured with the Austrian health insurance (ÖGK). Since 01/01/2020, Austria has reduced the number of social insurance institutions to five (ÖGK, SVS, BVAEB, PVA, AUVA).

A social insurance institution in Austria helps the insurer in case of health problems and collects money for his/her pension. Accordingly, the insurance institutions collect money from their insurer and use it to pay some bills. The social insurance contribution is regulated like a tax on income. Along with the fact that social insurance institutions have to pay tax to the government, these are the transactions that are implemented in this research. The process how of social insurance institutions work is not part of the research in this thesis.

The social insurance rate is about 40% of the defined basis assessment (monthly earnings), of which a part is paid by the employer and a part is deducted from the employee's salary. The maximal defined basis assessment is 5,370 € per month.

Employee share for social insurance = 18.12 %

Employer share for social insurance = 21.23 %

For the sake of convenience, not all five social insurances are implemented in the project, although the code provides interfaces to do so.

3.1.3 Individual person

In case of employment, many taxes in Austria are carried out between enterprises and the government, which means that an individual person does not has as many transactions in the system. Neither way an individual person can buy goods or services from an enterprise to trigger tax events and needs an overview of his personal income tax calculation. Furthermore, they are beneficiary of the social insurance institution therefore the customer of them. The implemented agent is a "user-like" agent and is focused on simple usability to support availability for many individuals. A government cannot expect financial expertise from every individual person in Austria.

3.1.4 Enterprise

Enterprises are the economic participants with the most ingoing and outgoing transactions. In this thesis, commercial companies are mainly considered because they are more involved in tax issues than voluntary service institutions or other non-financial enterprises. Besides the tax regulation already explained in the government description, enterprises interact with all other participants or even other enterprises in conducting their business, selling products or services. Besides calculating VAT on each transaction, a sales history is needed for some tax calculations. This implementation assumes that all transactions are operated within the Ethereum network and without hard cash. This is a major political barrier but it is not important for a feasibility study.

By reason that enterprises can exist with many different characteristics, the implementation is a simplification of the entire reality with the possibility to extend. Sole proprietorship, partnership (OG, KG), corporation (GmbH, AG) and other corporate forms are treated equally.

3.1.5 Capital collecting point

In this thesis, capital collecting points, investment associations and other insurance companies than social insurance will not be considered for the implementation, for reasons of simplification.

In the model, a capital collecting point is an entity that saves or lends money from other business participants. They earn interest for savings or have to pay interest for lent money. The bank is responsible for capital return tax. Because transactions are no longer executed by banks, the capital collecting points do not play such an important role in the model but need to exist for money-lending systems and all other financial products. In further research, it could be investigated how capital return tax could be levied directly from the individual and how a blockchain system could eventually replace individual processes of our current bank system. Because this thesis investigates how to automate the actual economy system, capital collecting points remains in the model and the impacts on them are discussed. If financial transactions were invariably executed by the blockchain, financial institutions could focus more on other products and services like money lending and fund investments.

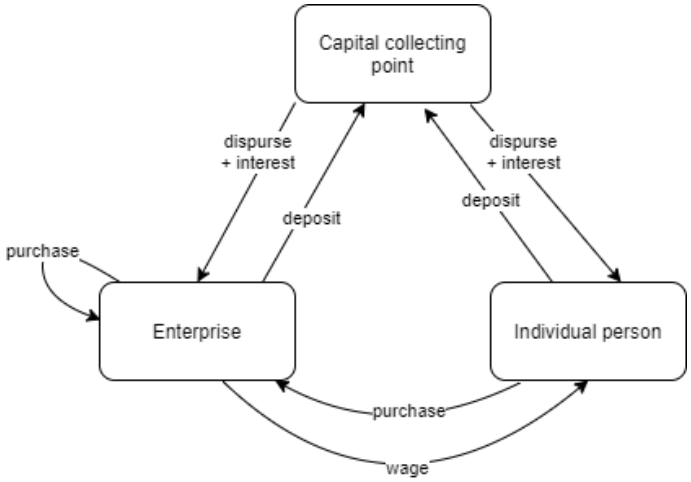


Figure 3.2: Basic transaction model

3.2 Economic interactions

In this chapter, the identified transactions are grouped by seven different transaction groups and modeled separately. The models are created from the view of the money flow. Accordingly, an arrow from an individual person to an enterprise represents a money flow from a person to a company, for example. It is important to mention that one single entity in real life is able to appear in different roles in our model depending on the actual interaction. A bank is a capital collection point in case of savings and interests, but it is an enterprise in case of selling other services or receiving services. This model defines roles that can be occupied by real-life entities for automating their transactions. Identification of single participants and fraud control (tax evasion) is not the goal of this thesis. Its assumed that an enterprise account is an enterprise in real life. In further research, identification methods and security issues should be discussed.

3.2.1 Basic transactions

First, the basic transactions that are necessary for calculating taxes are modeled. Capital collection points collect money from the other business participants and pay interest on the money collected. Loans are not part of the model because they are not needed for simple tax calculations. In this master thesis, banking is simplified to the most rudimentary function and is not implemented in the smart contract. Decentralized finance [con] is a separate topic with already-implemented solutions in the Ethereum environment.

Enterprises pay wages to individual persons, which are the basis for income tax of individuals.

Enterprises and individual persons purchase products or services from business entities.

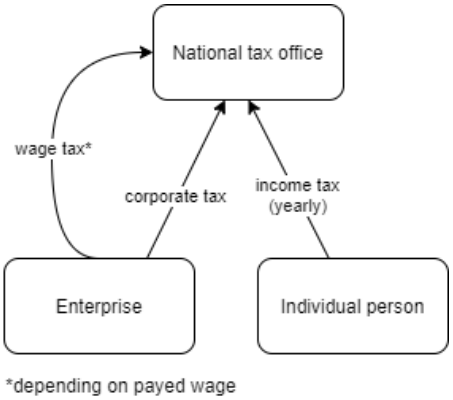


Figure 3.3: Transaction model income taxes

3.2.2 Income taxes

The calculation of the income tax depends on the income type. Wages are taxed directly, which means that enterprises have to pay wage tax for each employed individual. Enterprises have to pay corporate tax based on their yearly profits. The profit or income is estimated in the first instance and divided into a monthly fee. The total amount is corrected with the tax adjustment in the following year. The tax adjustment also applies to individual persons for reporting special spending or earnings. With a direct calculation of tax, this master thesis aims to supersede yearly tax adjustments.

3.2.3 Value added tax

VAT is added to each transaction and forwarded from the vendor to the tax office. Enterprises are able to recall their VAT expenses. VAT is a consumer tax, and therefore it only has to be paid by individual persons. The automated tax service should be able to forward and recall the VAT automatically without any intervention from the participants.

3.2.4 Social insurance

Similar to income tax, social insurance fee is based on the income of employees. The model is simplified because this master thesis does not aim to depict the total degree of social insurance companies in Austria. There are many differences between social insurance companies in Austria, but the fee system is very similar. The fees are all paid to an entity based on wages and respective income.

Because social insurance is subject to insurance tax, a portion of insurance fees are transferred to the tax office.

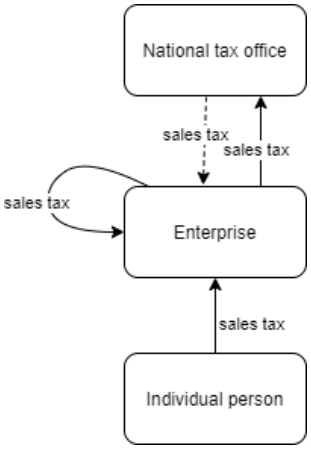


Figure 3.4: Transaction model VAT

Figure 3.5: Transaction model social insurance

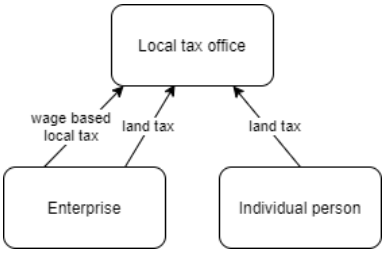


Figure 3.6: Transaction model local tax

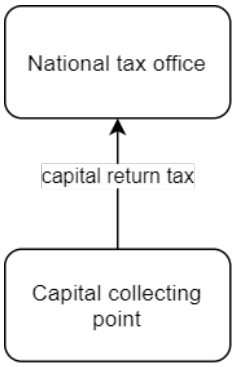


Figure 3.7: Transaction model capital return tax

3.2.5 Local tax

The local tax office is responsible for local tax based on wages and land tax based on property. While local tax can be calculated automatically with wage transactions, land tax has to be calculated separately because it depends on an external factor. Like individual taxes, the land tax calculation is not part of the experiment but a possibility to transfer the tax amount is present.

3.2.6 Capital return tax

Capital return tax is processed between the tax office and the capital collecting point. It is removed from the interests and redirected by the bank.

3.2.7 Individual taxes

Finally, some taxes for particular industry sectors are left. These taxes are calculated based on external values like alcohol or oil production or shipping. Because implementing interfaces for all kinds of special taxes exceeds the scope of this thesis, the amount of individual taxes has to be calculated externally. The system provides a transaction with the tax amount as an input to record the expanse.

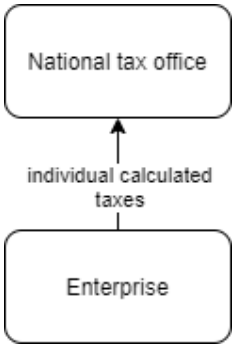


Figure 3.8: Transaction model individual taxes

3.3 Class diagram

In the next step, a class diagram is developed with the findings of the transaction models. The programming systematics of the Solidity programming language are considered in the creation of the class diagram. To reduce transaction fees, the amount of transactions is kept small. Transact every small tax amount to the tax office would lead to disproportionate transaction fees. Smart contracts are able to store ether without an additional transaction. Therefore, the tax amount is kept on the smart contract and an authority collects it afterwards. In the case of multiple tax offices (national, local), the contract has to ensure that only the authorized entity is able to collect the associated tax money. A detailed description of each class is presented in the implementation part of the thesis.

3.4 Limitations of the model

This model is a simplified meta model of the Austrian economy and it does not represent every detail of each entity. However, the basic concepts of how tax is paid from business entities and individual persons to the national tax office, local tax office and social insurance are implemented entirely to prove the technical possibilities.

Furthermore, trade between country borders is not included in the project at all, thus assuming that the economy is a closed system without the need for external interfaces. This is important for calculation of taxes. For example, income tax is calculated based on all income transferred through the system. Income outside the system (cash) cannot be considered by the calculation method. This would require some input methods to manually change the income value, which is not the intention of an automated tax system. Besides trust issues, this makes such projects politically very disputable because governments would have to force everybody to use it.

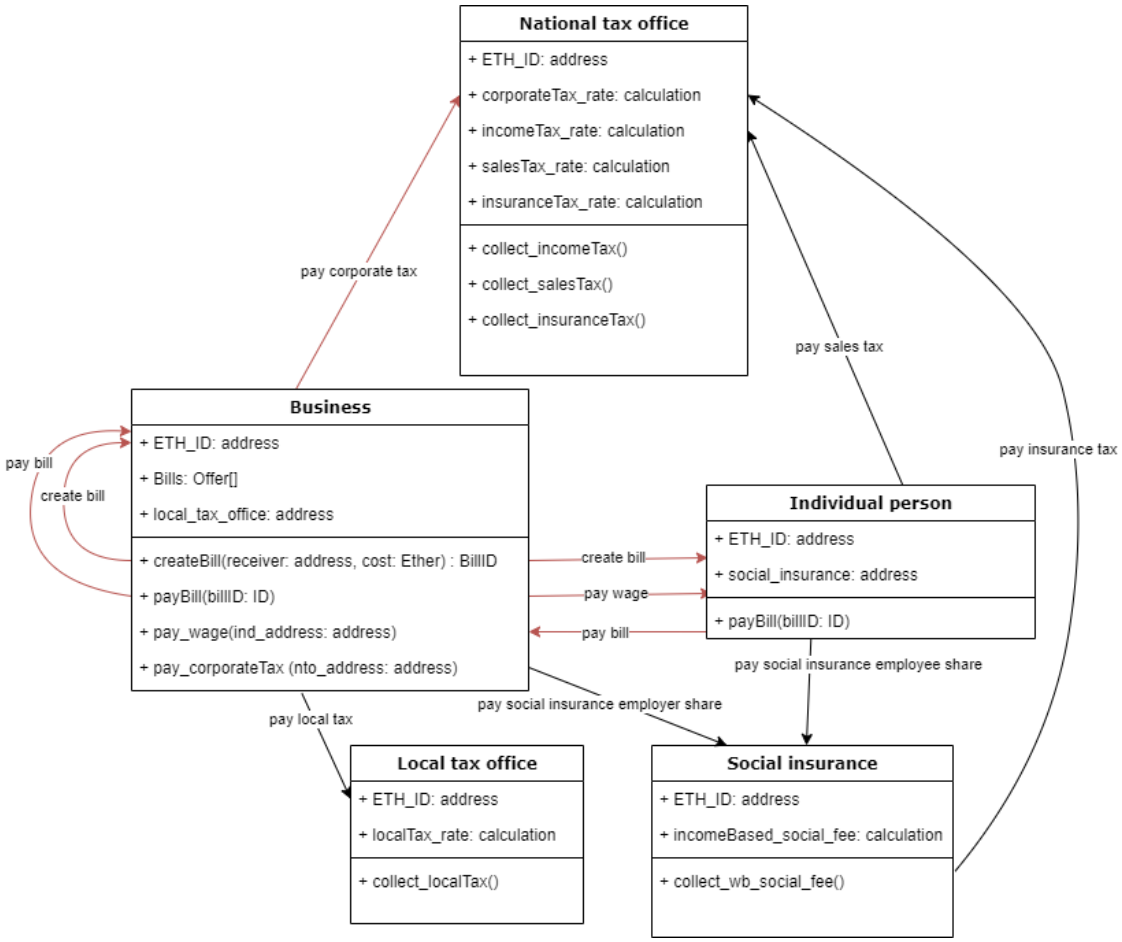


Figure 3.9: Class diagram

CHAPTER 4

Implementation – Smart contract

The smart contract is written on the Ethereum blockchain because it has the most developing tools and is supported by a broad development community. Only a few technologies available on Ethereum that are used for this master thesis are discussed. Ethereum is a global open-source platform with multiple program languages, development environments and a wide range of tools, but it is in an early stage of development.

4.1 Technologies

Solidity is the most popular program language for Ethereum. It is an object-oriented program language inspired by JavaScript. Solidity is compiled to bytecode that is executable on the Ethereum virtual machine.

Remix IDE is a browser-based IDE for Solidity dApps. It is used to write and compile the Solidity code and it is able to deploy the contract and run transactions based on the contract functions.

http://remix.ethereum.org

Ganache is a tool to create a local Ethereum blockchain for testing purposes.

https://www.trufflesuite.com/ganache

Etherscan is used for code verification. The code deployed of a contract ID can be viewed and verified by users.

https://kovan.etherscan.io/address/0x52B0a2531Bd462b67D9c959AC0Ac7c17f2851717#code

4.2 Concept

First of all, programming a smart contract means special discipline for the developer. It has to be clear that data stored in objects will be stored on the blockchain and changing

anything will require a chargeable transaction. The more data that is stored, the higher the price for the transaction. Furthermore, the smart contract needs to be deployed on the blockchain with a chargeable transaction based on the length of the compiled code. Once deployed on main net, the smart contract will be accessible as long as the Ethereum network exists and it cannot be deleted or changed. The only way to "update" a smart contract is to deploy a new one and tell people to use the new contract address. Due to the transactions with money, a smart contract should not be able to fall into an unstable state or give any possibility to misuse the functionality.

The implementation is divided into multiple contracts, which always implement all previous contracts. As a result, the order of the contracts is important: if a contract calls a function from another contract, it has to be in one of the previous contracts. This means that the basic structure of the implementation has to be in the first contracts and the more complex functions come later.

4.3 Code description

4.3.1 Libraries

For safe integer calculations, the library *safemath* is added to the contract at the beginning. It provides functions to add, subtract, divide or multiply integer values and rejects faulty transactions due to integer overflow. The blockchain would start to count from zero again after an overflow that is fatal with array IDs.

BokkyPooBahsDateTimeLibrary is used for date calculations. Taxes strongly depend on our calendar, but Ethereum only works with UNIX timestamps. The functions in this library are able to convert UNIX timestamps to calendar dates, and vice versa. Given that UNIX to calendar date conversion is not trivial due to leap years, the advantage of a community-driven open-source platform provides the possibility to import these functions.

4.3.2 Contract ownable

In our scenario, the smart contract is owned by the national tax office, which means that there are functions that are only executable by the address of the national tax office. Accordingly, the smart contract has to be ownable by an address. In the constructor, the ownership of the smart contract is set to the sender address, namely the deploy address. In this contract, it is not possible to change the owner address to another Ethereum key, and the deploy address will always remain the contract owner. The onlyOwner modifier can be used in other functions to prevent non-owner addresses from using it. Public view functions are user-created to view the data stored in the blockchain. In this case, the owner address is public for every Ethereum address.

```
1  constructor () internal {
2    _owner = msg.sender;
```

```
3     emit OwnershipTransferred(address(0), _owner);}
```

Listing 4.1: Contract ownable

4.3.3 Structs

This contract represents the entire data structure for all following functions. It defines all objects stored on the blockchain. To optimize transaction costs, mappings are used to implement the dependencies between objects and store information about addresses.

First, the objects that are save later in a list are defined. The bill object represents the data of a business transaction. Accountants usually collect this information for tax calculations of their customers. In our case, the data of a bill is stored on the blockchain and can be used for tax calculations by the contract itself. A bill has an amount, a VAT percentage, a date and a Boolean value concerning whether it has already been paid or not. Objects to store the base value of income per year, corporate income per year and the base value of the monthly insurance calculation are created. Essentially, the monthly insurance object stores the gross value of income and the yearly income object the net value of income.

```
1     struct Bill {
2         uint256 amount;
3         uint32 salesTax;
4         uint256 date;
5         bool paid;
6     }
7
8     struct YearlyIncome {
9         uint year;
10        uint256 incomeSum;
11    }
12
13    struct CorporateIncome {
14        uint year;
15        int256 incomeSum;
16        bool paid;
17    }
18
19    struct MonthlyInsurance {
20        uint year;
21        uint month;
22        uint256 insuranceSum;
23    }
```

Listing 4.2: Struct objects

Furthermore, two global variables for the sum of insurance fees and local taxes stored on the contract are defined. If social insurance fees are collected from wage transactions, the taxes are not directly directed to the social insurance; instead, they are kept on the contract and can be collected by social insurance any time. The amount of total social

insurance fees and local taxes can only be collected from the responsible social insurance or local tax office. Not even the contract owner is able to collect those ethers.

Now the lists to store multiple objects in the smart contract are created. To add an object to a list, a chargeable transaction on the Ethereum blockchain is necessary. This is essentially how to define the database in a smart contract. Objects stored in this list will last there and can be only changed by other contract function (internal). Therefore, developers have to be particularly careful with functions accessing this list. The safemath library is especially for list operations, because an integer overflow would point to an incorrect ID in the array list.

```
1    YearlyIncome[] internal yearlyIncome;
2
3    CorporateIncome[] internal corporateIncome;
4
5    MonthlyInsurance[] internal monthlySocialInsurance;
```

Listing 4.3: Object lists

In this implementation, the contract owner determines which Ethereum public keys are individual persons, businesses, social insurances or local tax offices. This information is saved in a mapping from the address to a Boolean value. Moreover, an individual person has a relation to a social insurance address and a business address to a local tax office address. For each social insurance address and local tax office address, a sum of the collectible ether is stored.

Finally, the references from addresses to the listed objects are implemented. A bill has an owner address and a receiver address. For list operations, the counts of bills of an owner address and receiver address are stored. The other list objects each have a referring address, as well as the count for list operations.

4.3.4 Modifier

A modifier can be used in functions to restrict access to them. Functions for business addresses, social insurance addresses and local tax offices are defined. Furthermore, the access to some functions for a specific bill ID is restricted. To prevent functions from encountering an error due to an invalid bill ID, the modifier isValidBill checks this.

```
1    modifier onlyBusinessAddress() {
2        require (businessAddresses[msg.sender] == true, "Only business
             addresses can access this function");
3        _;
4    }
5
6    modifier onlySocialInsuranceAddress() {
7        require (socialInsuranceAddresses[msg.sender] == true, "Only social
             insurance addresses can access this function");
8        _;
9    }
10
```

```
11      modifier onlyLocalTaxOfficeAddress() {
12          require (localTaxOfficeAddresses[msg.sender] == true, "Only tax
                office addresses can access this function");
13          _;
14      }
15
16
17
18      modifier onlySeller(uint _billID) {
19          require (billToOwner[_billID] == msg.sender, "Only bill owner can
                access this function");
20          _;
21      }
22
23      modifier onlyBillReceiver(uint _billID) {
24          require (billToReceiver[_billID] == msg.sender, "Only bill reveiver
                can access this function");
25          _;
26      }
27
28      modifier isValidBill(uint _billID) {
29          require(_billID < bills.length, "Invalid bill ID");
30          _;
31      }
```

Listing 4.4: Modifier

4.3.5 Contract owner

This contract contains all functions which can be only executed by the contract owner, the tax office. The contract provides views to show the different amounts of ether stored on the contract. The total contract balance, the total insurance fee balance and the total local tax balance.

The tax office is able to withdraw the total contract balance excluding social insurance fees and local taxes.

```
1       function contract_withdraw() external onlyOwner {
2           address payable _owner = contract_owner();
3           _owner.transfer(address(this).balance.sub(socialInsuranceFeeSum.add(
                localTaxOfficeSum)));
4       }
```

Listing 4.5: Withdraw balance

Furthermore, the creation of the different address types is defined. To add a business address, the address cannot already be an individual person address and a valid local tax office address need to be chosen. A social insurance and a local tax office address cannot already be an individual person address. Finally, an individual person address may not already be a business, social insurance or local tax office address, and a valid social insurance address has to be added to the individual person address.

The withdraw functions for social insurance and local tax office addresses are implemented. Social insurance and local tax office addresses are able to withdraw the taxes collected from their customers. In case of social insurance, a fraction of the collected sum is paid to the tax office as insurance tax. The rest is transferred to the address of the social insurance or local tax office.

To allow the public to view the type of an address, public functions are defined.

4.3.6 Corporate tax

Corporate tax contracts are the first contracts for tax calculations. Corporate tax is calculated on a yearly basis depending on the revenue of a business address. Accordingly, the bills paid to the business address are added to the corporate income sum, paying wage and bills is subtracted from the corporate income sum. 25% of the total income sum has to be paid to the tax office. This also means that the corporate tax can only be calculated for past years and therefore it can only be paid after a year has passed.

```
1    function payCorporateTax(uint _corporateIncomeID) external  payable
         onlyBusinessAddress{
2        require(corporateIncome[_corporateIncomeID].paid == false);
3        require(corporateIncomeToReferringAddress[_corporateIncomeID] == msg.
             sender);
4        require(BokkyPooBahsDateTimeLibrary.getYear(block.timestamp) >
             corporateIncome[_corporateIncomeID].year);
5
6        //Calculate value for transaction
7        require(msg.value == _calculateCorporateTax(_corporateIncomeID) * 1
             wei, "Not enough value in the transaction");
8
9        corporateIncome[_corporateIncomeID].paid = true;
10   }
11
12   function _calculateCorporateTax(uint _corporateIncomeID) internal view
         returns(uint256){
13       uint corporateTaxRate  = 2500;  //25,00%
14       if(corporateIncome[_corporateIncomeID].incomeSum <= 0){
15           return 0;
16       }else{
17           uint256 returnValue =  uint256(corporateIncome[_corporateIncomeID
                 ].incomeSum);
18           return returnValue.div(100).mul(corporateTaxRate).div(100);
19       }
20   }
```

Listing 4.6: Corporate tax calculation

The referring corporate income year is detected by the actual pay date of ingoing or outgoing transactions. If no corporate income year object exists for the current year, it is created. The history of past year will still be visible.

For the table views of corporate income years, functions to show the content of the corporate income list are defined.

4.3.7 Social insurance

Social insurance fees are calculated based on monthly income of an individual person address. The fee comprises an employer and an employee share and it has a monthly maximum basis assessment of 5,370€. The sum of all wage transactions in a month is stored in the monthly social insurance income sum. In case the total sum is above 5,370€, no more social insurance fees are subtracted from or added to the wage. The employee share is added to the necessary transaction value of the outgoing wage transaction and the employer share is subtracted from the incoming transaction value to the individual person address. For example, if an employer creates a transaction to pay 1 ether to a person, he has to pay 1.2123 ethers for the transaction and the employee would receive 0.8188 ethers in his wallet. The rest is stored on the contract and can be collected by the referring social insurance address.

```
1    function _socialInsuranceFeeCalculation(uint256 _incomeSum, uint256
         _grossValue) internal pure returns(uint256, uint256){
2        uint employeeShareFee  = 1812;  //18,12%
3        uint employerShareFee  = 2123;  //21,23%
4        uint256 employeeShare = 0;
5        uint256 employerShare = 0;
6        uint256 maxBasisAssessment = 25000000000000000000; //25 ether
7        if(_incomeSum < maxBasisAssessment){
8            if(_grossValue.add(_incomeSum) <= maxBasisAssessment ){
9                employeeShare = _grossValue.div(100).mul(employeeShareFee).
                     div(100);
10               employerShare = _grossValue.div(100).mul(employerShareFee).
                     div(100);
11           }
12           else{
13               employeeShare = maxBasisAssessment.sub(_incomeSum).div(100).
                     mul(employeeShareFee).div(100);
14               employerShare = maxBasisAssessment.sub(_incomeSum).div(100).
                     mul(employerShareFee).div(100);
15           }
16       }
17       return (employeeShare ,employerShare);
18   }
19
20   function _calculateSocialInsuranceFee( address _receiverAddress, uint256
         _grossValue) internal returns( uint256, uint256){
21       uint[] memory referringMonthlySocialInsuranceList =
             _getMonthlySocialInsuranceByReceiver(_receiverAddress);
22       (bool currentSocialInsuranceIDExist, uint currentSocialInsuranceID) =
              _getCurrentMonthlySocialInsurance(
             referringMonthlySocialInsuranceList);
23       if(currentSocialInsuranceIDExist == false){
24           currentSocialInsuranceID = _createMonthlySocialInsurance(
                 _receiverAddress);
```

```
25          }
26          (uint256 employeeShare, uint256 employerShare) =
                _socialInsuranceFeeCalculation(monthlySocialInsurance[
                currentSocialInsuranceID].insuranceSum, _grossValue);
27          monthlySocialInsurance[currentSocialInsuranceID].insuranceSum =
                monthlySocialInsurance[currentSocialInsuranceID].insuranceSum.add
                (_grossValue);
28          return (employeeShare, employerShare);
29      }
```

Listing 4.7: Social insurance calculation

The current social insurance month is detected by the date of the wage transaction. If no social insurance month object for the referring individual person address exists, a new one is created.

For the table views of social insurance months, functions to show the content of the social insurance month list are defined.

4.3.8 Income tax

Income tax is calculated based on the yearly income of an individual person address. The income tax rate increases with higher income. For simplification reasons, only three income levels are implemented, even if there are more in reality. Income from 0 ether to 0.5 ethers is tax free, while income above 0.5 and below 1 ether is taxed at 25%. From 1 to 2 ethers, the income tax is 35%, and the income tax for everything above 2 ethers is 50%. This leads to a higher total tax rate compared to reality but makes it easier to test even with smaller amounts of ether. All incoming wage transactions are considered as income tax events and the tax is directly deducted from the transaction value.

```
1       function _calculateIncomeTax( address _receiverAddress, uint256
            _grossValue) internal returns( uint256){
2           uint[] memory referringYearlyIncomeList =  _getyearlyIncomeByReceiver
                (_receiverAddress);
3           (bool currentIncomeYearExist, uint currentIncomeID) =
                _getCurrentIncomeYear(referringYearlyIncomeList);
4           if(currentIncomeYearExist == false){
5               currentIncomeID = _createYearlyIncome(_receiverAddress);
6           }
7           uint256 netValue = _taxCalculation(yearlyIncome[currentIncomeID].
                incomeSum, _grossValue);
8           yearlyIncome[currentIncomeID].incomeSum = yearlyIncome[
                currentIncomeID].incomeSum.add(_grossValue);
9           return netValue;
10      }
11
12      function _taxCalculation(uint256 _incomeSum, uint256 _grossValue) private
             pure returns(uint256){
13          uint256 netValueLevel0 = 0;
14          uint256 incomeLevel1 = 500000000000000000;
15          uint256 incomeLevel2 = 1000000000000000000;
```

```
16          uint256 incomeLevel3 = 2000000000000000000;
17          uint256 netValueLevel1 = 0;
18          uint256 netValueLevel2 = 0;
19          uint256 netValueLevel3 = 0;
20          uint256 newIncomeSum = _incomeSum.add(_grossValue);
21
22          //Income level 0 and 1
23          if(newIncomeSum > incomeLevel1){
24              if( incomeLevel2 > _incomeSum){
25                  if(_incomeSum >= incomeLevel1){
26                      if(newIncomeSum < incomeLevel2){
27                          netValueLevel1 = _grossValue.div(100).mul(75);
28                      }else{netValueLevel1 = incomeLevel2.sub(_incomeSum).div
                            (100).mul(75);
29                      }
30                  }else{
31                      netValueLevel0 = incomeLevel1.sub(_incomeSum);
32                      if(newIncomeSum < incomeLevel2){
33                          netValueLevel1 = _grossValue.sub(netValueLevel0).div
                                (100).mul(75);
34                      }else{netValueLevel1 = incomeLevel2.sub(incomeLevel1).div
                            (100).mul(75);
35              }}}
36          }else{netValueLevel0 = _grossValue;}
37
38          //Income level 2
39          if(newIncomeSum > incomeLevel2){
40              if( incomeLevel3 > _incomeSum){
41                  if(_incomeSum >= incomeLevel2){
42                      if(newIncomeSum < incomeLevel3){
43                          netValueLevel2 = _grossValue.div(100).mul(65);
44                      }else{netValueLevel2 = incomeLevel3.sub(_incomeSum).div
                            (100).mul(65);
45                      }
46                  }else{
47                      if(newIncomeSum < incomeLevel3){
48                          netValueLevel2 = newIncomeSum.sub(incomeLevel2).div
                                (100).mul(65);
49                      }else{netValueLevel2 = incomeLevel3.sub(incomeLevel2).div
                            (100).mul(65);
50          }}}}
51
52          //Income level 3
53          if(newIncomeSum > incomeLevel3){
54              if( incomeLevel3 > _incomeSum){
55                  netValueLevel3 = newIncomeSum.sub(incomeLevel3).div(100).mul
                        (50);
56              }else{netValueLevel3 = _grossValue.div(100).mul(50);
57          }}
58          return netValueLevel0 + netValueLevel1 + netValueLevel2 +
                netValueLevel3;
59      }
```

Listing 4.8: Income tax calculation

The current income year is detected by the date of the wage transaction. If no income year object for the referring individual person address exists, a new one is created.

For the table views of income years, functions to show the content of the yearly income list are defined.

4.3.9 Local tax

Local tax is added to wage transactions based on the gross value similar to the employer share of social insurance. The added amount is 3% of the gross transaction value.

```
1    function _calculateLocalTaxAmount( uint256 _grossValue) internal pure
         returns( uint256){
2        uint localTaxPercent = 3;  //3%
3        uint256 localTaxAmount = _grossValue.div(100).mul(localTaxPercent);
4        return localTaxAmount;
5    }
```

Listing 4.9: Local tax calculation

4.3.10 Wage

The pay wage function combines all tax functions. The local tax and employer share of social insurance are calculated and added to the required transaction value. Furthermore, the employee share and income tax are subtracted from the transferred value. This function is for business addresses to pay wages to individual person addresses.

```
1    function payWage(address payable _receiverAddress, uint256 _grossValue)
         external payable onlyBusinessAddress returns( uint256) {
2        require(individualPersonAddresses[_receiverAddress] == true);
3
4        //Calculate social insurance
5        (uint256 employeeShare, uint256 employerShare) =
             _calculateSocialInsuranceFee(_receiverAddress, _grossValue);
6        //Calculate local tax
7        uint256 localTaxAmount = _calculateLocalTaxAmount(_grossValue);
8        //Required value is raised by social insurance employee share and
             local tax
9        require(msg.value == _grossValue.add(employerShare).add(
             localTaxAmount) * 1 wei, "Not enough value in the transaction");
10       //Calculate net value of transaction (income tax and social insurance
              employee share)
11       uint256 netValue = _calculateIncomeTax(_receiverAddress, _grossValue.
             sub(employeeShare));
12
13       //Add social insurance fees to social insurance sums
14       address socialInsuranceAddress = individualPersonToSocialInsurance[
             _receiverAddress];
15       uint256 socialInsuranceFee = employeeShare.add(employerShare);
16       socialInsuranceAddressSum[socialInsuranceAddress] =
             socialInsuranceAddressSum[socialInsuranceAddress].add(
             socialInsuranceFee);
```

```
17          socialInsuranceFeeSum = socialInsuranceFeeSum.add(socialInsuranceFee)
                ;
18
19          //Add local tax to local tax office sums
20          address localTaxOfficeAddress = businessToLocalTaxOffice[msg.sender];
21          localTaxOfficeAddressSum[localTaxOfficeAddress] =
                localTaxOfficeAddressSum[localTaxOfficeAddress].add(
                localTaxAmount);
22          localTaxOfficeSum = localTaxOfficeSum.add(localTaxAmount);
23
24          //Reduce corporate tax income by total paid amount
25          _subCorporateTaxValue(msg.sender, _grossValue.add(employerShare).add(
                localTaxAmount) * 1 wei);
26
27          _receiverAddress.transfer(netValue * 1 wei);
28
29          return netValue;
30      }
```

Listing 4.10: Pay wage

4.3.11 Bills

Business addresses are able to create bills for a receiving address with an amount, a VAT rate and a receiver address.

```
1       function  _createBill (uint256 _amount, uint32 _salesTax, address
            _receiver)  private returns(uint)  {
2           //Increase array length
3           bills.length = bills.length.add(1);
4           uint billID = bills.length.sub(1);
5           //Insert bill data
6           bills[billID].amount = _amount;
7           bills[billID].salesTax = _salesTax;
8           bills[billID].date = now;
9           bills[billID].paid = false;
10
11          //Mapping to bill owner
12          billToOwner[billID] = msg.sender;
13          //Mapping to bill receiver
14          billToReceiver[billID] = _receiver;
15
16          //Increase bill counts
17          ownerBillCount[msg.sender] = ownerBillCount[msg.sender].add(1);
18          receiverBillCount[_receiver] = receiverBillCount[_receiver].add(1);
19
20          return billID;
21      }
```

Listing 4.11: Pay bill

Only the receiver address can pay the bill. The VAT is calculated depending on the type of receiver address. If the receiver is a business address, no VAT is added to the

transaction; otherwise, the VAT is added to the bill amount, stored on the contract and can be collected by the tax office. It is never transferred to the bill owner.

For the table views of bills, functions to show the content of the bill list are defined.

4.3.12 Individual taxes

Individual taxes are not implemented in the deployed contract due to a length limit of contracts. This kind of taxes would be calculated outside of the system and paid directly to tax office, local tax office or social insurance. Furthermore the contract could implement interfaces for adding transactions outside of the system. If a bill is payed cash, the business address would have to add the value for tax calculation without any ether transaction.

CHAPTER 5

Implementation – Website

To access the smart contract functions, a website with some sort of Ethereum wallet implementation is necessary. In fact, there are multiple ways to access the functions even without a website. This thesis is focused on a simple implementation with JavaScript and MetaMask to show the possibilities of the implemented smart contract.

5.1 Technologies

Bootstrap is a free, front-end CSS framework for faster and easier web development. It includes HTML-and CSS-based design templates for typography, forms, buttons, tables, navigation, modals, image carousels and many others, as well as optional JavaScript plugins. The sb-admin template is used to design the website. [Stab]

MetaMask is a browser extension that acts as a bridge between internet browsers, Ethereum and decentralized applications built on the Ethereum network. It enables users to execute Ethereum dApps in their internet browser directly without running a full Ethereum node. MetaMask allows users to store, send, receive, and facilitate interactions with the Ethereum network. MetaMask is a web browser extension for Google Chrome, Opera, Firefox, and Brave. [gol]

JavaScript is used to connect to an Ethereum wallet via MetaMask and implement the functions in the bootstrap template.

5.2 Concept

Essentially a simple website with tables to view the data on the smart contract and forms to execute functions with input values is necessary. Besides the introduction page, all other pages need a connected MetaMask wallet to be accessible. If MetaMask is not yet

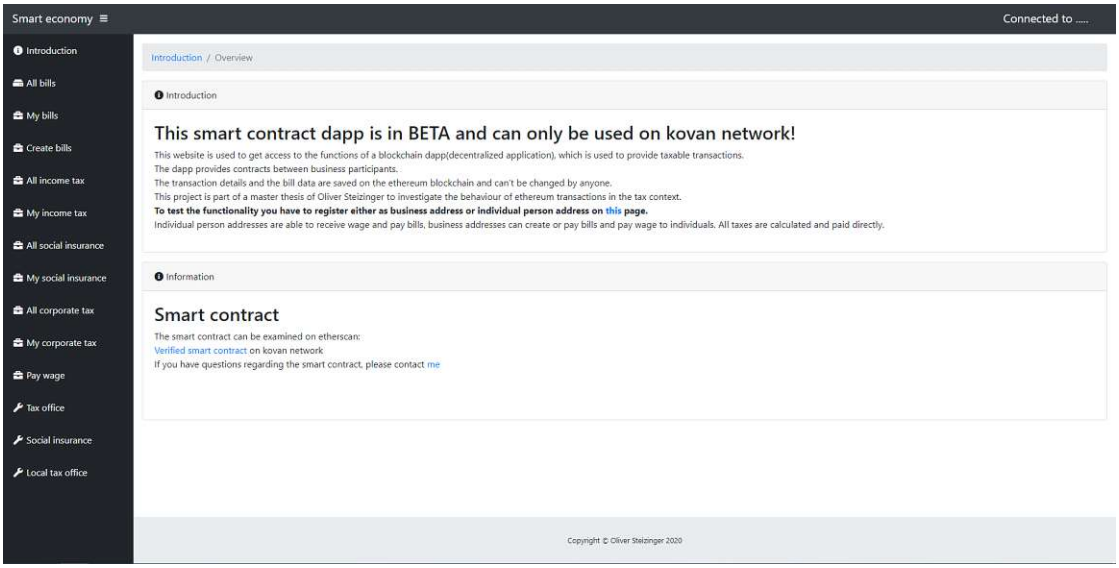


Figure 5.1: Website

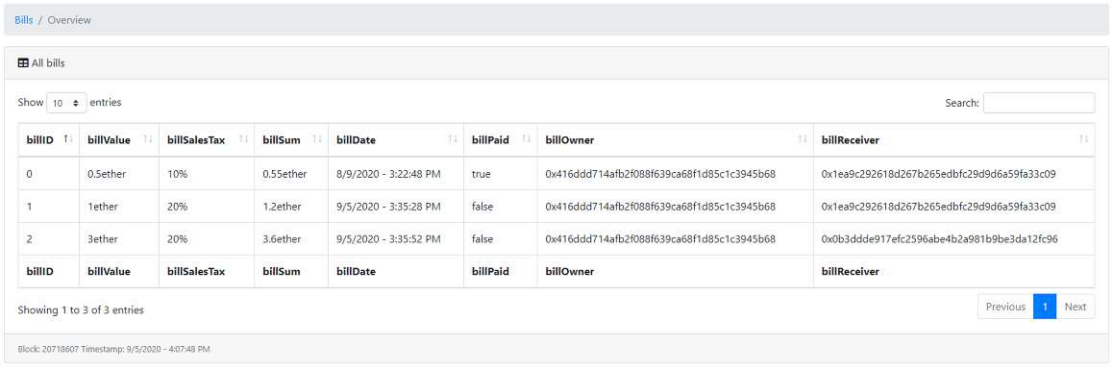


Figure 5.2: All bills

installed, the page will lead the user to the MetaMask download page. The views and functions are shown on thirteen pages.

Website link: http://www.smarteconomy.at

5.2.1 All bills

The 'all bills' view shows all bills in the list stored on the smart contract. It includes all properties of a bill and the bill ID. As the bills are public, anyone with a connection to the Ethereum network is able to view the bills.

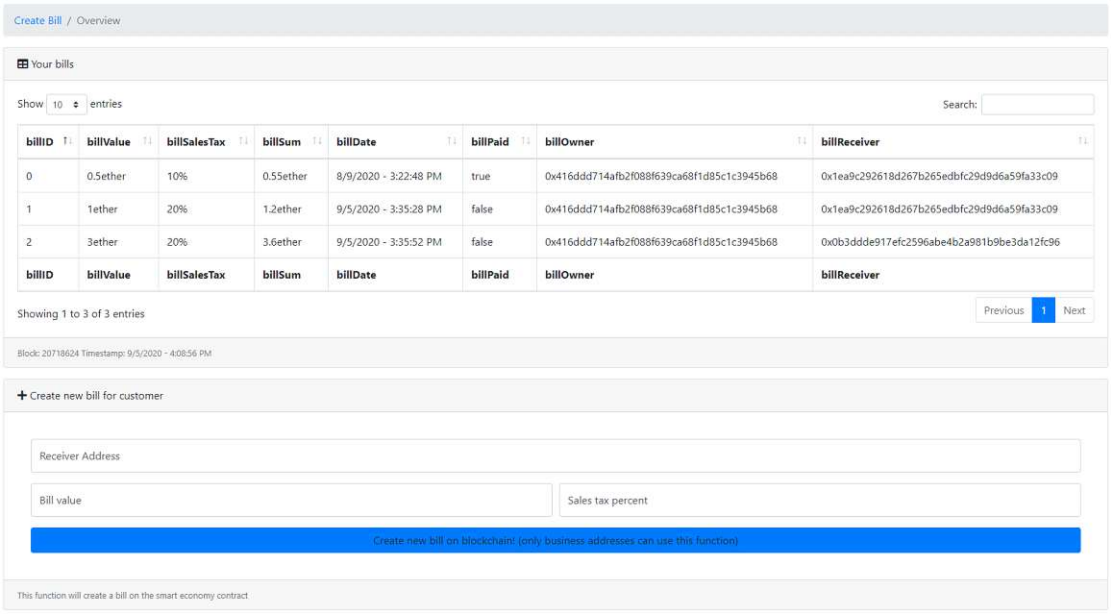


Figure 5.3: Create bills

5.2.2 My bills

'My bills' essentially shows the same data as the 'all bills' view but filters it by the receiver. This means that only the received bills of the connected address are displayed. Furthermore, the user is able to pay his unpaid bills with a button to generate a transaction.

5.2.3 Create bills

On this page, business addresses can see their created bills and create new bills. To create a bill, users have to insert a receiver address, a gross value of the product sold and the VAT percentage.

5.2.4 All income tax / My income tax

The tables in the income tax pages shows the yearly income objects stored on the blockchain. With the ID, the year, the current income sum and the referring individual person address are displayed. The only difference between "All income tax" and "My income tax" is that the latter only shows the data of the currently-connected address.

5.2.5 All social insurance / My social insurance

On these pages, the monthly income sum for social insurance calculations is displayed. The ID, year, month, gross value of income and the referring address are shown. Only

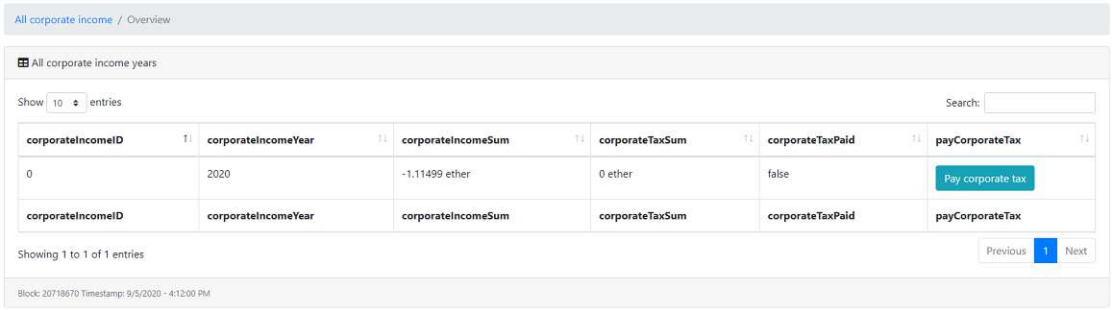


Figure 5.4: My corporate tax

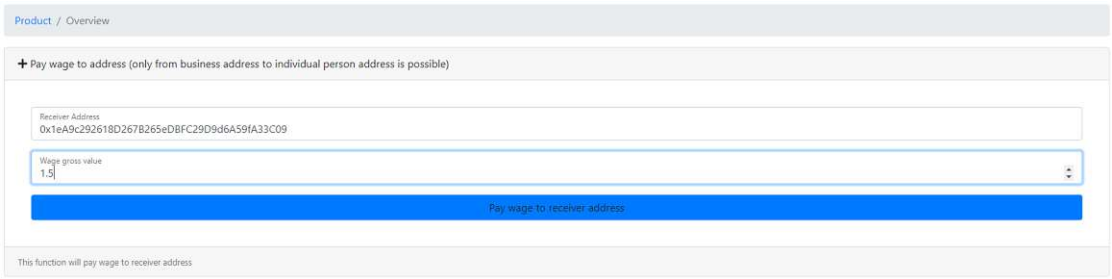


Figure 5.5: Pay wage

individual person addresses have social insurance months. The “My social insurance” page only shows months of the connected Ethereum address.

5.2.6 All corporate tax

For business addresses, the total corporate income is stored during a year. This page shows all business addresses with a corporate income. ID, year, total income sum and the referring business address are displayed in the table. A negative sum means that the business address had more expenditures than earnings.

5.2.7 My corporate tax

Corporate income tax is calculated with the yearly income, which means that it is possible to calculate the total sum at earliest after a year ends. This is implemented into the pay transaction in the table. A business address can only pay corporate taxes of past years.

5.2.8 Pay wage

With this function, business addresses are able to pay wage to an individual person address with all tax calculations included. The user has to insert a receiver address and a gross value. The function calculates the employer share and adds it to the transaction value. Furthermore, the employee share is deducted from the transferred amount.

Figure 5.6: Contract balance

5.2.9 Tax office

The following functions enable the tax office (contract owner) to create new addresses to participate in the system. Social insurance addresses and local tax office addresses can be created without any other information besides the public Ethereum address.

A business address always needs a valid local tax office address for local tax purposes. Business addresses cannot be constructed without a referring local tax office.

Individual person addresses always need a valid social insurance address.

Finally, the contract owner is able to view the total contract balance, the social insurance share and the local tax share. The total contract balance reduced by the social insurance share and local tax share can be transferred to the contract owner address with the withdraw contract balance function.

5.2.10 Social insurance

Social insurance addresses can view and withdraw their social insurance fees received on this page.

5.2.11 Local tax office

Local tax office addresses can view and withdraw their local taxes received on this page.

5.3 Code description

In this section, the JavaScript code to unite the smart contract with the webpages is described. The smart contract file implements the connection between MetaMask and the smart economy contract. Furthermore, function files for each page are implemented.

5.3.1 smartContract

Upon page load, MetaMask is connected to the smart contract, if this has not already taken place on a previous page.

The checkMetaMask function checks whether MetaMask is running and it triggers an access request to allow the website to use a MetaMask wallet. If the request is successful, the connection is established.

```
async function checkMetaMask() {
      // Modern dapp browsers...
      if (window.ethereum) {
        window.web3 = new Web3(ethereum);
        try {
          // Request account access if needed
          await ethereum.enable();
          // Acccounts now exposed
          connectMetaMask();

        } catch (error) {
          // User denied account access...
          window.location = "/html_error/error_metamask_login.html";
        }
      }
      // Legacy dapp browsers...
      else if (window.web3) {
        window.web3 = new Web3(web3.currentProvider);
        // Acccounts always exposed
        connectMetaMask();
      }
      // Non-dapp browsers...
      else {
                window.location.href = "/html_error/error_metamask_install.
                    html";
        console.log('Non-Ethereum browser detected. You should consider
            trying MetaMask!');
      }
}
```

Listing 5.1: Check MetaMask

With MetaMask running, the web3 account can be used to create the smart contract.

Contract ID is loaded and an interval for a check whether MetaMask is still running is implemented.

To select the correct contract ID, the network currently connected to MetaMask has to be identified. With this information, the referring contract ID stored in a global variable is redirected to the contract function.

```
function setContractID() {
    web3.version.getNetwork((err, netId) => {
        switch (netId) {
            case "1":
```

```
                document.getElementById("conntectedTo").innerHTML = '
                    Connected to mainnet';
                console.log('Connected to mainnet');
                setContract(0);
                break
            case "2":
                document.getElementById("conntectedTo").innerHTML = '
                    Connected to the deprecated Morden test network';
                console.log('Connected to the deprecated Morden test network.
                    ');
                setContract(0);
                break
            case "3":
                document.getElementById("conntectedTo").innerHTML = '
                    Connected to the ropsten test network';
                console.log('Connected to the ropsten test network.');
                setContract(contractIDRopstenNetwork);
                break
            case "4":
                document.getElementById("conntectedTo").innerHTML = '
                    Connected to the Rinkeby test network';
                console.log('Connected to the Rinkeby test network.');
                setContract(0);
                break
            case "42":
                document.getElementById("conntectedTo").innerHTML = '
                    Connected to the Kovan test network';
                console.log('Connected to the Kovan test network.');
                setContract(contractIDKovanNetwork);
                break
            default:
                document.getElementById("conntectedTo").innerHTML = '
                    Connected to an unknown network';
                console.log('Connected to an unknown network.');
                setContract(contractIDLocalNetwork);
        }
    });
}
```

Listing 5.2: Set contract ID

The functions of the contract are defined in a separate file called ABI, which is generated by the Ethereum compiler. With the ABI and the contract ID, the connection is established and the contract functions can be used with JavaScript.

Finally, data for the current page is loaded with page-specific functions.

```
function loadData() {
    var url = window.location.pathname;
    var filename = url.substring(url.lastIndexOf('/') + 1);
    if (filename == "index.html") {
            //no data to load
    } else if (filename == "main_bills.html") {
        loadBillData();
```

```
    } else if (filename == "main_myBills.html") {
        loadMyBillData();
    } else if (filename == "main_createBill.html") {
        loadBillData();
    } else if (filename == "main_taxOffice.html") {
            //no data to load
    } else if (filename == "main_payWage.html") {
            //no data to load
    } else if (filename == "main_myIncomeTax.html") {
        loadMyIncomeData();
    } else if (filename == "main_mySocialInsuranceFees.html") {
        loadMySocialInsuranceData();
    } else if (filename == "main_allIncomeTax.html") {
        loadAllIncomeData();
    } else if (filename == "main_allSocialInsuranceFees.html") {
        loadAllSocialInsuranceData();
    } else if (filename == "main_allCorporateTax.html") {
        loadAllCorporateIncomeData();
    } else if (filename == "main_myCorporateTax.html") {
        loadMyCorporateIncomeData();
    }
}
```

Listing 5.3: Load data

In the following page functions, the dltConnection object is used to call the functions that are defined in the smart contract.

5.3.2 billFunctions

First, all bill IDs from the blockchain with the getAllBills function are read. For each bill ID, all variables are loaded.

```
function displayBills(_billIDs) {
    billIDs = _billIDs;
    var listLength = billIDs.length;
    dataTableData = [];
    itemsProcessed = 0;
    billIDs.forEach(function (billID) {
        dltConnection.getBillValue(billID, function (error, result) {
            var billValue = result;
            dltConnection.getBillSalesTax(billID, function (error, result) {
            var billSalesTax = result;
            dltConnection.getBillSum(billID, function (error, result) {
            var billSum = result;
            dltConnection.getBillDate(billID, function (error, result) {
            var billDate = result;
            dltConnection.getBillPaid(billID, function (error, result) {
            var billPaid = result;
            dltConnection.getBillOwner(billID, function (error, result) {
            var billOwner = result;
            dltConnection.getBillReceiver(billID, function (error, result) {
            var billReceiver = result;
```

```
21              addDataObjects(error, billID, billValue, billSalesTax, billSum,
                    billDate, billPaid, billOwner, billReceiver, listLength);
22          });});});});});});}});
23      });
24  }
```

Listing 5.4: Bill functions

Data objects are added to the HTML table.

5.3.3 myBillFunctions

This implementation is similar to billFunctions but it only shows bills with the connected wallet as the bill receiver. Furthermore, a pay bill button is added, which executes the payBill smart contract function.

5.3.4 createBillFunctions

This implementation is similar to billFunctions, but it only shows bills with the connected wallet as the bill owner. Furthermore, business addresses can create new bills with the createBill smart contract function.

```
1  function createNewBill() {
2      var inputReceiverAddress = document.getElementById('inputReceiverAddress'
           ).value;
3      var inputBillValue = document.getElementById('inputBillValue').value;
4      var inputTaxPercent = document.getElementById('inputTaxPercent').value;
5
6      if (isNumeric(inputBillValue) && isNumeric(inputTaxPercent) &&
           inputReceiverAddress != "") {
7          var billValue_wei = inputBillValue * 1000000000000000000;
8          dltConnection.createBill(billValue_wei, inputTaxPercent,
               inputReceiverAddress, { from: userAccount }, function (error,
               result) {
9              if (error) console.log(error);
10         });
11     }
12 }
```

Listing 5.5: Create Bill functions

5.3.5 allIncomeFunctions / myIncomeFunctions

The income pages are views only and they display all income objects or only the income objects of the connected wallet.

5.3.6 allSocialInsuranceFunctions / mySocialInsuranceFunctions

The social insurance pages are view-only and they display all social insurance objects or only the social insurance objects of the connected wallet.

5.3.7 allCorporateTaxFunctions

This page is a view-only and it displays all corporate income objects.

5.3.8 myCorporateTaxFunctions

This implementation is similar to allCorporateTaxFunctions, but it only shows corporate income from the connected wallet. Furthermore, a pay corporate tax button is added, which executes the pay corporate tax smart contract function for a previous year.

```
function payCorporateTax(_incomeID, _corporateTaxValue) {
    dltConnection.payCorporateTax(_incomeID, { from: userAccount, value:
        _corporateTaxValue }, function (error, result) {
        if (error) console.error(error);
    });
}
```

Listing 5.6: Pay corporate tax

5.3.9 payWageFunctions

The pay wage function executes the smart contract function for wage transactions.

```
function payWage() {
    var inputReceiverAddress = document.getElementById('inputReceiverAddress'
        ).value;
    var inputWageValue = document.getElementById('inputWageValue').value;

    if (isNumeric(inputWageValue) && inputReceiverAddress != "") {
        var wageValue_wei = inputWageValue * 1000000000000000000;

        dltConnection.getSocialInsuranceEmployerShare(wageValue_wei,
            inputReceiverAddress,  function (error, result) {
            var wageValueEmployerShare_wei = result;
            dltConnection.getLocalTaxAmount(wageValue_wei, function (error,
                result) {
                var wageValueLocalTax_wei = result;
                var wageValueNet_wei = (wageValue_wei * 1) + (
                    wageValueEmployerShare_wei * 1) + (wageValueLocalTax_wei
                    * 1);
                console.log(wageValueNet_wei);

                dltConnection.payWage(inputReceiverAddress, wageValue_wei, {
                    from: userAccount, value: wageValueNet_wei }, function (
                    error, result) {
                    if (error) console.error(error);
                });
            });
        });

    }
}
```

Listing 5.7: Pay wage function

5.3.10 taxOfficeFunctions

The tax office function allows the contract owner to check and add business, social insurance, local tax office and individual person addresses.

```
function addBusinessAddress() {
    var inputBusinessAddress = document.getElementById('inputBusinessAddress').value;
    var inputLocalTaxOfficeAddress = document.getElementById('inputReferringLocalTaxOfficeAddress').value;

    dltConnection.addBusinessAddress(inputBusinessAddress, inputLocalTaxOfficeAddress, { from: userAccount}, function (error, result) {
            if (error) console.error(error);
        });
}

function isBusinessAddress() {
    var inputIsBusinessAddress = document.getElementById('inputIsBusinessAddress').value;
    dltConnection.getBusinessAddressState(inputIsBusinessAddress, { from: userAccount }, function (error, result) {
        if (error) console.error(error);
        var isBusinessAddress = result;
        if (isBusinessAddress == true) {
            document.getElementById('inputIsBusinessAddress').style.backgroundColor = "green";
        } else {
            document.getElementById('inputIsBusinessAddress').style.backgroundColor = "red";
        }
    });
}
```

Listing 5.8: Business address functions

The contract owner is able to view contract balances and withdraw the tax office share of the contract balance.

```
function getContractBalance() {
    dltConnection.contract_getContractBalance({ from: userAccount }, function (error, result) {
        if (error) console.error(error);
        var contractBalance_ether = result / 1000000000000000000;
        document.getElementById('contractBalance').value = contractBalance_ether + " ether";
    });
}

function getInsuranceFeeBalance() {
    dltConnection.contract_getSocialInsuranceFeeBalance({ from: userAccount }, function (error, result) {
        if (error) console.error(error);
```

```
12          var contractBalance_ether = result / 1000000000000000000;
13          document.getElementById('socialInsuranceFeeBalance').value =
                contractBalance_ether + " ether";
14      });
15  }
16
17  function getLocalTaxBalance() {
18      dltConnection.contract_getLocalTaxOfficeBalance({ from: userAccount },
            function (error, result) {
19          if (error) console.error(error);
20          var contractBalance_ether = result / 1000000000000000000;
21          document.getElementById('localTaxBalance').value =
                contractBalance_ether + " ether";
22      });
23  }
24
25  function withdrawBalance() {
26      dltConnection.contract_withdraw({ from: userAccount }, function (error,
            result) {
27      });
28  }
```

Listing 5.9: Balance functions

5.3.11 socialInsuranceFunctions

Social insurance addresses can view and withdraw their share of the contract balance.

5.3.12 localTaxOfficeFunctions

Local tax offices can view and withdraw their share of the contract balance.

CHAPTER 6

Tests and emulation

6.1 Test data description

To test the smart contract, we use the latest data of the Austrian statistic office "Statistik Austria" from 2018.

6.1.1 Business participant numbers

The business participants comprise one tax office, multiple local tax offices, social insurances, businesses and individuals. These entities can be quantified from the Austrian statistics, whereby in 2018 Austria had **8,822,267** residents. Not all residents are directly business participants and need a registered address. A bill can be paid with an unregistered address. Only if a business wants to skip VAT it does need a registered address.

To receive a wage or salary, a person needs an **individual person address** to calculate social insurance fees. To create a new individual person address, the tax office has to register an address and set a social insurance address, which is responsible for the

Year	2018
Residents	**8,822,267**
Employees	**3,044,226**
Companies	**346,469**
Tax office	**1**
Social insurances	**5**
Local tax offices	**2095**

Table 6.1: Business participant numbers

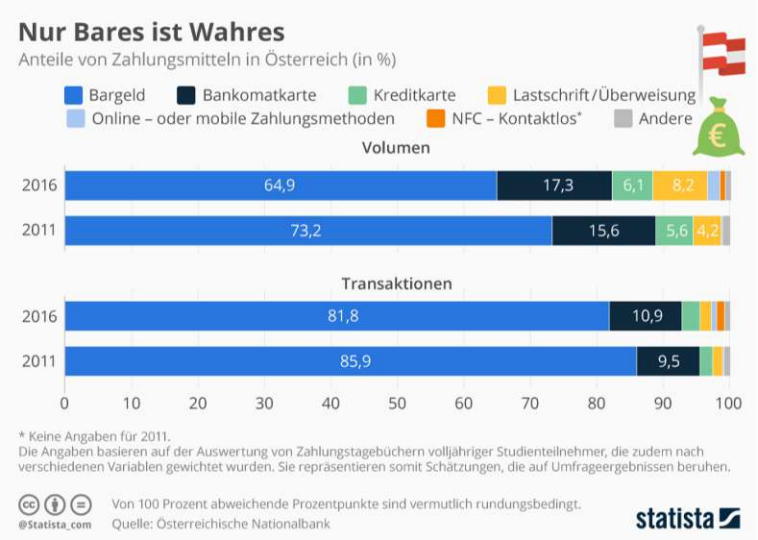


Figure 6.1: Transactions in Austria [Moh]

individual and receive the fees. Therefore, every employed person in Austria would need such an address. In 2018, the average number of employed persons was **3,044,226** people.

To pay a wage or salary, create bills and pay tax-reduced bills, a company has to be registered as a business address. To register an address as a business, a responsible local tax office address has to be set. In 2018, the total number of companies in Austria was **346,469**.

The contract owner address represents the tax office. There is only one tax office in Austria, which collects sales, income and corporate tax.

Each individual person address is connected to a social insurance address, which collects the social insurance fee. Following the consolidation of the social security agencies, there are currently **five** in existence.

For local taxes, a business address has to be connected to a local tax office address. Each local community in Austria is represented by one local tax office address. There are **2,095** local communities in Austria.

6.1.2 Transactions

The smart contract provides the five key functions of the defined economy model. These transactions are relevant for tax calculation and collection. Therefore, data for these transactions in the Austrian economy is needed to assess the capabilities of the current state of development. The functions are pay wage, create bill, pay bill, pay corporate tax, and withdraw taxes.

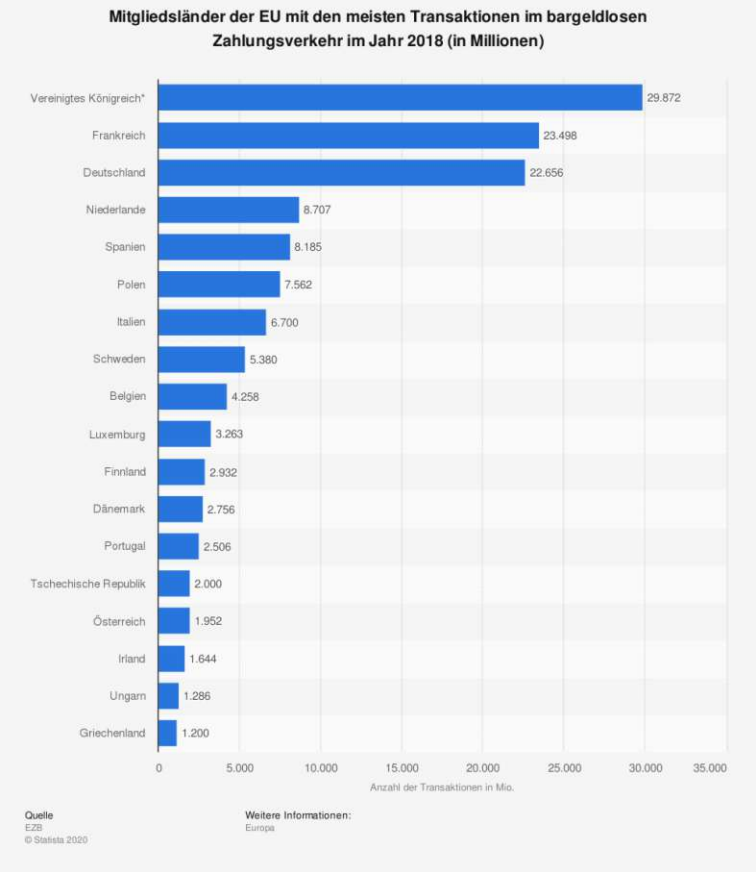


Figure 6.2: Transactions in Europe [staa]

Pay wage is usually paid monthly, whereby one transaction per employee and month can be assumed. With 3,044,226 employees, this makes **36.53 million** transactions per year.

Because Statistik Austria does not provide data about the number of transactions, the number of cashless transactions and the proportion compared to cash transactions is used for the calculation. With **1.95 billion** cashless transactions and a share of 20%, the estimated total number of transactions per year is **9.75 billion** transactions in Austria.

Corporate tax has to be paid by a company once a year, making **346,469** such transactions per year.

Because the taxes are collected as a total sum by a few tax entities, these transactions have only little influence. If tax collection points collect taxes once a month, there will be about **100** transactions per year.

6.2 Test environment

Blockchain systems are well suited for emulation purposes because for test purposes it

Transaction	Estimation
Pay wage	**36,530,712**
Create bill (cashless only)	**1,952,000,000**
Pay bill (cashless only)	**1,952,000,000**
Create bill (total transactions)	**9,760,000,000**
Pay bill (total transactions)	**9,760,000,000**
Pay corporate tax	**346,469**
Withdraw taxes	**100**

Table 6.2: Transaction numbers

does not make a difference if the blockchain runs on multiple machines or a local test environment. The limitation of block size and block time is equal regardless of whether the system runs on one or multiple machines. "Speeding up" the chain is limited to the computing power of the local machine. The minimum block time of the Ganache client is 1 second, which is about fifteen times faster than the original Ethereum blockchain.

The following tools are used to run a local Ethereum blockchain and run automated transactions on the smart contract.

Ganache is a tool to create a local Ethereum blockchain for testing purposes.

https://www.trufflesuite.com/ganache

Visual Studio Code is a code editor with blockchain extensions.

https://code.visualstudio.com

Truffle Suite is a development framework for smart contracts. It provides automated contract testing, which allows programming and executing contract functions in JavaScript.

https://www.trufflesuite.com/truffle

Due to limitations of computing power, time and tool support, the tests focus on following objectives:

- Is the current Ethereum blockchain able to carry out a sufficient number of transactions needed by the Austrian economy per year?
- Which modifications are necessary to test multiple year usage of the smart contract on a local network?
- Are smart contracts suitable to implement tax laws?

6.3 Test description

6.3.1 Basic function tests

Tests the contract with a small number of transactions to validate the functionality of the contract. The Ganache test chain can be set to automine for this test. With this setting, the blockchain creates a new block for every transaction without a time delay. Because the performance is not tested in this part, automining is used for faster test results. Furthermore, the test code awaits the result of the transaction call, which means that it is not possible to have multiple transactions in a single block.

Before the transactions can be executed, the smart economy environment has to be initialized. This means that the contract owner address (accounts[0]) proves other addresses to be able to use the smart contract functions. First, tax office and social insurance addresses are necessary for the creation of business and individual addresses.

```
1  it("Initialize smart contract environment - Create business address", async
       () => {
2    let instance = await SmartEconomyContract.deployed();
3
4    await instance.addBusinessAddress(accounts[4],localTaxOfficeAddress, {
         from: accounts[0]});
5    var isBusinessAddress = await instance.getBusinessAddressState(accounts
         [4], {from: accounts[4]});
6    businessAddress= accounts[4];
7
8    assert.equal(isBusinessAddress, true);
9    console.log("Log: Business address created - getBusinessAddressState
         returns " + isBusinessAddress);
10 });
```

Listing 6.1: Create business address

In each test, the transaction is executed first, and the taxes are withdrawn from tax entities afterwards to check whether the correct amount was transferred.

The pay wage function transfers wage from a business address to an individual person address and collects income tax, social insurance fees, local tax and thereby also insurance taxes via the social insurance address. The insurance tax is deducted from the withdrawable social insurance fee balance of a social insurance address.

```
1  it("Execute transactions - Pay wage", async () => {
2    let instance = await SmartEconomyContract.deployed();
3
4    var wageValue_wei = 1000000000000000000n;
5    var wageValueEmployerShare_wei = await instance.
         getSocialInsuranceEmployerShare(wageValue_wei.toString(),
         individualPersonAddress, {from: businessAddress});
6    var wageValueLocalTax_wei = await instance.getLocalTaxAmount(
         wageValue_wei.toString(), {from: businessAddress});
7    var wageValueNet_wei = wageValue_wei + BigInt(wageValueEmployerShare_wei)
          + BigInt(wageValueLocalTax_wei);
```

```
8
9       var transactionData = await instance.payWage(individualPersonAddress,
            wageValue_wei.toString(), {from: businessAddress, value:
            wageValueNet_wei.toString()});
10
11      assert.equal(transactionData['receipt']['status'], true);
12      console.log("Log: Transaction pay wage sent. Transaction value = " +
            wageValueNet_wei + " wei");
13  });
```

Listing 6.2: Pay wage

The bill function is tested with an individual receiver and a business receiver because they have to pay different prices. An individual receiver has to pay VAT, while a business address does not.

```
1   it("Execute transactions - Create bill to individual person", async () => {
2     let instance = await SmartEconomyContract.deployed();
3
4     var billValue_wei = 1000000000000000000n;
5     var inputTaxPercent = 20;
6
7     var transactionData = await instance.createBill(billValue_wei.toString(),
           inputTaxPercent.toString(), individualPersonAddress, { from:
          businessAddress });
8
9     assert.equal(transactionData['receipt']['status'], true);
10    console.log("Log: Bill created from business address to individual person
          ");
11  });
```

Listing 6.3: Create bill

```
1   it("Execute transactions - Pay bill individual person address", async () =>
        {
2     let instance = await SmartEconomyContract.deployed();
3
4     //Includes sales tax percent
5     var billValue_wei = 1200000000000000000n;
6     var _billID = 0;
7
8     var transactionData = await instance.payBill(_billID, { from:
          individualPersonAddress, value: billValue_wei.toString() });
9
10    assert.equal(transactionData['receipt']['status'], true);
11    console.log("Log: Bill paid by individual person address - value = " +
          billValue_wei + " wei");
12  });
```

Listing 6.4: Pay bill

After the bills have been created, the receiver pays them and the VAT is directed to the tax office.

6.3.2 Test environment limitation tests

Tests the limitations of the test environment by increasing the number of transactions. The test chain is set to the minimum block time of one second. Some functions of the smart contract need to read the transaction value from a smart contract view, which slows down the transaction creation. Therefore, hardcoded values are used for the transaction values to create more transactions in a shorter time. The local testing machine is limited by the processing time of function calls. The processing time is divided into multiple machines (nodes) in the decentralized main net of Ethereum. Therefore, the limitation of computing power is a limitation for the test environment and different to the live network.

```
1  it("Execute transactions - Pay wage", async () => {
2    let instance = await SmartEconomyContract.deployed();
3    const Contract = new web3.eth.Contract(instance.abi, instance.address);
4
5    var wageValue_wei = "1000000000000000000";
6
7    for (let i = 10; i < 30; i++) {
8     var wageValueNet_wei = 1242300000000000000n;
9      Contract.methods.payWage(accounts[i+30], wageValue_wei).send({from:
           accounts[i], value: wageValueNet_wei.toString(), gas: 575000});
10     console.log("Log: Pay wage from: " + accounts[i] + " to: " + accounts[i
           +30]  +  " wei: " + wageValueNet_wei.toString());
11   }
12
13   console.log("Log: Pay wage finished");
14 });
```

Listing 6.5: Pay wages

```
1  it("Execute transactions - Create bills", async () => {
2    let instance = await SmartEconomyContract.deployed();
3    const Contract = new web3.eth.Contract(instance.abi, instance.address);
4
5    var billValue_wei = 10000000000000000000n;
6    var inputTaxPercent = 20;
7
8    for (let i = 10; i < 15; i++) {
9      Contract.methods.createBill(billValue_wei.toString(), inputTaxPercent.
           toString(), accounts[i+30]).send( { from: accounts[39-i], gas:
           375000 });
10     console.log("Log: Create bill from: " + accounts[39-i] + " to: " +
           accounts[i+30]  +  " wei: " + billValue_wei.toString() + " - " +
           inputTaxPercent + "% tax");
11   }
12
13   console.log("Log: Create bills finished");
14 });
```

Listing 6.6: Create bills

```
1  it("Execute transactions - Pay bills", async () => {
2    let instance = await SmartEconomyContract.deployed();
3    const Contract = new web3.eth.Contract(instance.abi, instance.address);
4
5    var billValue_wei_ind = 120000000000000000n;
6
7    for (let i = 0; i < 5; i++) {
8      var billReceiver = await instance.getBillReceiver(i, {from: accounts
           [0]});
9
10     Contract.methods.payBill(i).send({ from: billReceiver, value:
           billValue_wei_ind.toString(), gas: 555000});
11     console.log("Log: Pay bill individual receiver: " + billReceiver  +  "
           wei: " + billValue_wei_ind.toString());
12   }
13   console.log("Log: Pay bills individual finished");
14 });
```

Listing 6.7: Pay bills

Ganache test client is able to create 100 Ethereum addresses for test usage. We use this addresses for the limitation test. Furthermore, the transactions are called without awaiting the result, which allows the system to put multiple transactions into one block. To avoid errors due to unfinished transactions in a transaction chain, timeouts are used to slow down the transaction execution time.

The limitation test comprises the following series of transactions in a loop.

6.3.3 Gas consumption tests

Tests the gas consumption of the contract. An Ethereum block is limited by the amount of gas used by the transactions executed. To calculate how many transactions fit into one Ethereum block, the average gas consumption is necessary. Ethereum's block gas limit is 12,500,000.

For this purpose, single transaction calls are created to compare the transaction costs in the Ganache client.

6.3.4 Multiple year tests

In this test, the smart contract is modified to analyze how the contract behaves over several years. The "now" function of Solidity returns the current time. This function defines the pay dates that are used for further tax calculations. Therefore, the "now" function is replaced with a function that returns dates in the future instead of the current time. The future time is calculated with the number of seconds that have passed since the contract creation. The limitation test is adapted for the multiple year usage. This means that the multiple year test code is a collection of the previous functions and it can be viewed in the attached code.

```
1   function getGeneratedDate(uint currentTime) public view returns(uint date)
        {
2     uint timeDiff = currentTime.sub(contractCreationDate);
3     return contractCreationDate.add(timeDiff.mul(70000));
4   }
```

Listing 6.8: Modified date function

6.4 Test report

6.4.1 Basic function test

To check the results of the basic function test, we look at the results and recalculate the transaction amounts manually. All functions except the pay corporate tax function are executed at least once and taxes are withdrawn from the contract immediately afterwards. The pay corporate tax function is only accessible after a year has passed, which cannot be done in this test. With a fresh Ganache workspace, 100 new Ethereum addresses are set to a balance of 100 Ethereum to make changes of address balances comprehensible. The manually-calculated balances minus transaction costs have to be equal to the address balances after a basic function test run.

```
1   Contract: Smart economy basic function test
2 Log: Local tax office address created - getLocalTaxOfficeAddressState returns
      true
3     - Initialize smart contract environment - Create local tax office address
        (836ms)
4 Log: Social insurance address created - getSocialInsuranceAddressState
    returns true
5     - Initialize smart contract environment - Create social insurance address
        (1246ms)
6 Log: Individual person address created - getIndividualPersonAddressState
    returns true
7     - Initialize smart contract environment - Create individual person
       address (1262ms)
8 Log: Business address created - getBusinessAddressState returns true
9     - Initialize smart contract environment - Create business address (1126ms
       )
10 Log: Business address 2 created - getBusinessAddressState returns true
11    - Initialize smart contract environment - Create second business address
        (1020ms)
12 Log: Transaction pay wage sent. Transaction value = 1242300000000000000 wei
13    - Execute transactions - Pay wage (2008ms)
14 Log: Withdrawed tax office (income tax) = 79700000000000000 wei
15    - Execute transactions - Withdraw tax office balance (income tax) (1796ms
       )
16 Log: Withdrawed local tax office (local tax) = 30000000000000000 wei
17    - Execute transactions - Withdraw local tax office balance (local tax)
        (362ms)
18 Log: Withdrawed social insurance (social insurance fee) = 393500000000000000
    wei
```

```
19      - Execute transactions - Withdraw social insurance balance (social
            insurance fee) (483ms)
20  Log: Withdrawed tax office (insurance tax) = 5902500000000000 wei
21      - Execute transactions - Withdraw tax office balance (insurance tax) (855
            ms)
22  Log: Bill created from business address to individual person
23      - Execute transactions - Create bill to individual person (1234ms)
24  Log: Bill created from business address to business address 2
25      - Execute transactions - Create bill to business address (1143ms)
26  Log: Bill paid by individual person address - value = 1200000000000000000 wei
27      - Execute transactions - Pay bill individual person address (938ms)
28  Log: Withdrawed tax office (sales tax) = 200000000000000000 wei
29      - Execute transactions - Withdraw tax office balance (sales tax) (1721ms)
30  Log: Bill paid by business address - value = 1000000000000000000 wei
31      - Execute transactions - Pay bill business address (1237ms)
32  Log: Withdrawed tax office (zero) = 0 wei
33      - Execute transactions - Withdraw tax office balance (no sales tax
            business to business) (1883ms)
34
35
36    16 passing (23s)
```

Listing 6.9: Basic function test log

After the run, the transactions can be viewed in the Ganache client and the addresses values changed to a new state. Due to the automining property, there is only one transaction in each block. The balances of the addresses are changed to new values, which are compared with the manually-calculated values.

Manual calculation Pay wage 1 ETH

Employee share for social insurance (18.12% of gross value) = **0.1812 ETH**

Employer share for social insurance (21.23% of gross value) = **0.2123 ETH**

Insurance tax from social insurance to tax office (1.5% of employee + employer share) = **0.0059025 ETH**

Local tax for local tax office (3% of gross value) = **0.03 ETH**

Income tax (25% of 1 ETH minus employee share and 0.5 ETH tax free) – 0.3188/4 = **0.0797 ETH**

Manual calculation Pay bill individual 1 ETH

VAT (20% of price) = **0.2 ETH**

Manual calculation Pay bill business 1 ETH

No directly paid tax (but used for corporate tax calculation at end of year)

Comparing the results of the calculation and the balance of the addresses, it can be observed that the balances of the tax office address and the business address are too low. These missing values are the transaction costs for proving addresses in case of the tax

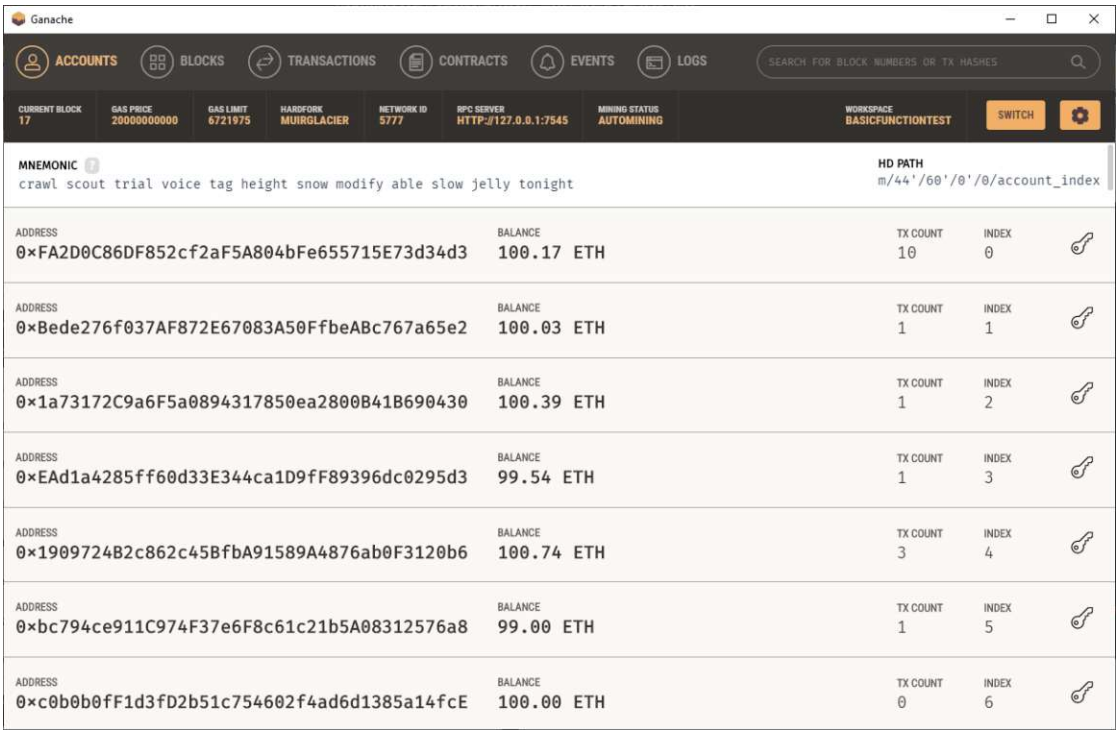


Figure 6.3: Ganache - Basic function test

ETH address Pay bill individual	Type Pay bill business	Start value End value	Contract creation	Pay wage
Index 0 +0.2	Tax Office	100 ETH **100.1784781 ETH**	-0.1071469	+0.2
Index 1	Local Office	100 ETH **100.03 ETH**		+0.03
Index 2	Insurance	100 ETH **100.3875975 ETH**		+0.3875975
Index 3 -1.2	Individual	100 ETH **99.5391 ETH**		+0.7391
Index 4 +1	Business +1	100 ETH **100.7577 ETH**		-1.2423
Index 5	Business -1	100 ETH **99 ETH**		

Table 6.3: Manual calculation

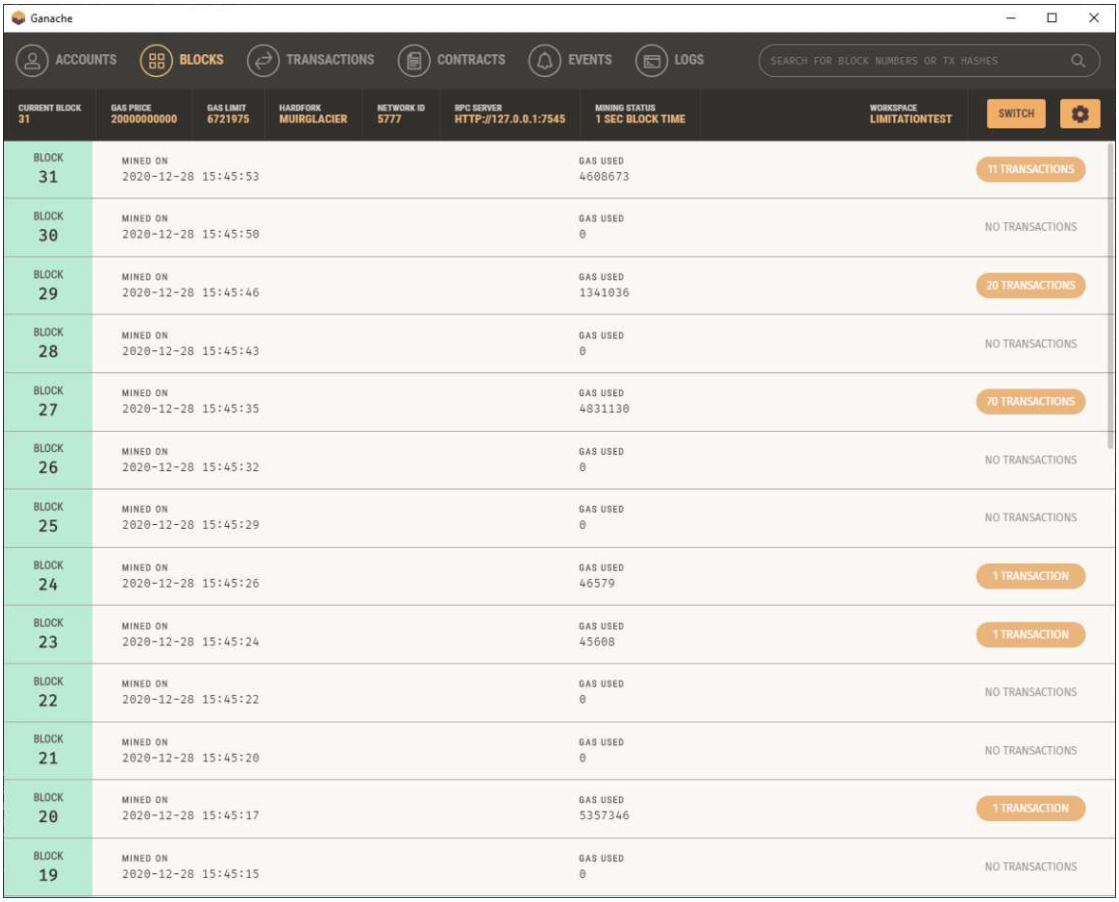


Figure 6.4: Ganache - Limitation test

office, paying wage and creating bills in case of the business address (0.01 ETH). Due to the high transaction costs of the contract creation, it has been added to the calculation. The contract is designed to spare individual person addresses from transaction costs as much as possible, which is the reason why the tax office and business addresses have to carry the most transaction costs. Only paying a bill is a charged transaction with a relatively low gas usage.

6.4.2 Limitation test

First, the limitation test is executed with two runs, and afterwards the number of runs is increased as far as possible to gain an idea of what the test environment is capable of. Due to the disabled automine function and the asynchrony transaction calls, multiple transactions per block are possible.

Looking at the timestamp of the blocks, it takes about 3 seconds for each block if transactions are executed, which is the limit for the test environment of this master

Figure 6.5: Limitation test error

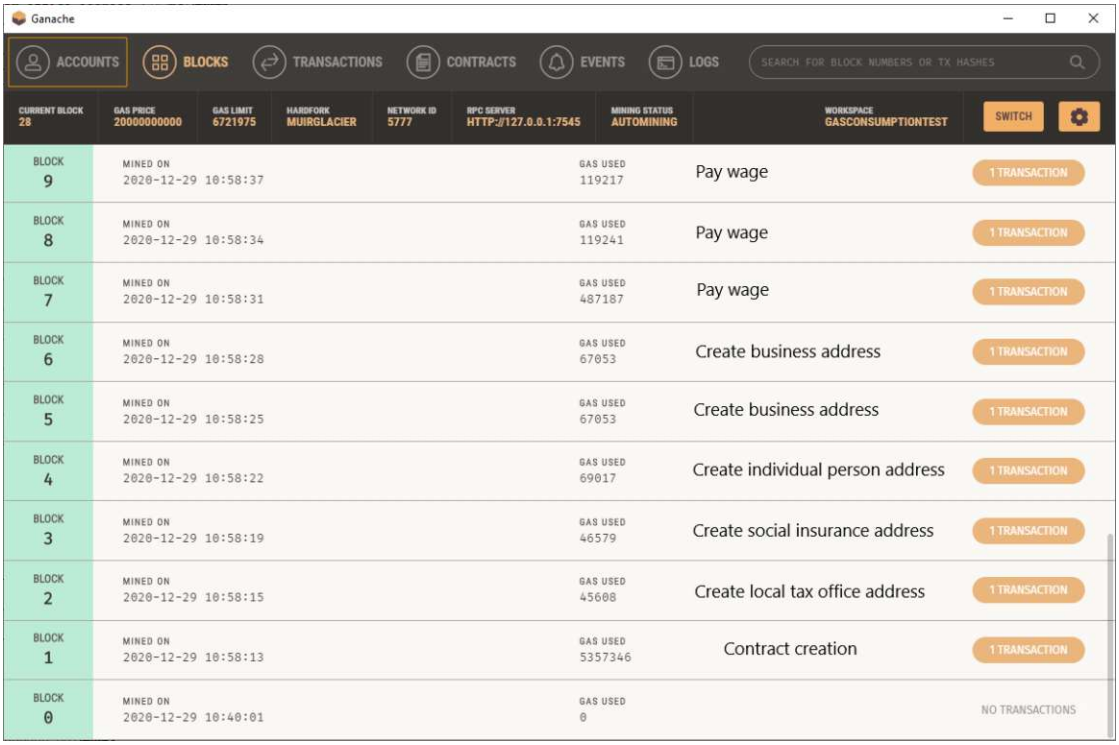


Figure 6.6: Ganache - Consumption test

thesis. Because the block time of the live Ethereum chain is between 10 and 20 seconds, the test environment is faster, but not able to emulate a much larger time frame in a shorter time than the original chain.

Increasing the number of runs leads to an error at a transaction call because the EVM cannot catch up with the incoming transaction calls. For the multiple year test, the number of transactions is reduced to obtain fewer transactions over a longer period of time.

6.4.3 Gas consumption test

For each function, the gas consumption is tested to ascertain how much computing power of the blockchain is necessary to run the smart contract. Because the contract creates storing objects for addresses per months or per year, varieties between multiple transaction calls are expected.

Gas price: 20000000000 wei (0.00000002 ETH)

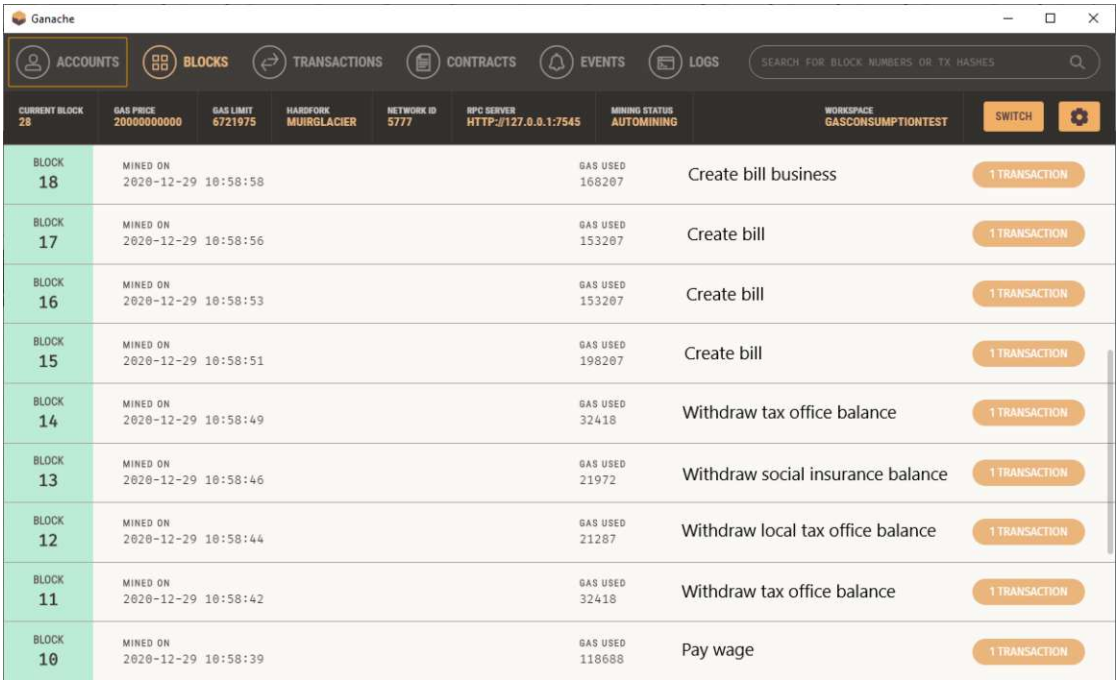


Figure 6.7: Ganache - Consumption test

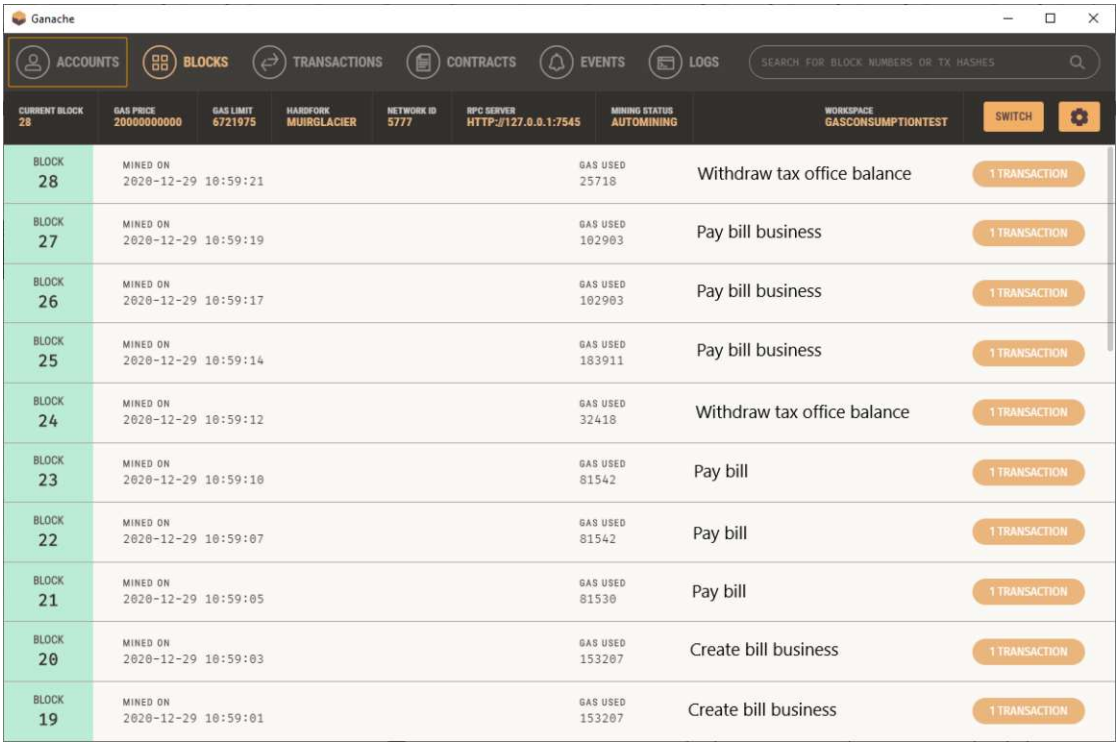


Figure 6.8: Ganache - Consumption test

Transaction	Gas used*	Costs*	Gas used	Costs
Contract creation	5357346	0.10715 ETH		
Create local tax office address	45608	0.00091 ETH		
Create social insurance address	46579	0.00093 ETH		
Create individual person address	69017	0.00138 ETH		
Create business address	67053	0.00134 ETH		
Pay wage	487187	0.00974 ETH	119000	0.00238 ETH
Create bill	198207	0.00396 ETH	153207	0.00306 ETH
Create bill business	168207	0.00336 ETH	153207	0.00306 ETH
Pay bill	81530	0.00163 ETH	81542	0.00163 ETH
Pay bill business	183911	0.00368 ETH	102903	0.00206 ETH
Withdraw tax office balance			32418	0.00065 ETH
Withdraw local tax office balance			21287	0.00043 ETH
Withdraw social insurance balance			21972	0.00044 ETH

Table 6.4: Gas usage

*The first transaction per receiver is more expensive due to object creation in the smart contract.

Because the pay wage transaction triggers the most tax events, it needs the most computing power and therefore consumes the most gas. In return, the other functions are relatively cheap in comparison with a simple transaction, which has a minimum gas limit of 21,000.

6.4.4 Multiple year test

To test the tax calculation over an entire year, the now function was replaced with a time multiplier of 70,000 and the waiting time in the limitation test is adjusted. After the test runs, the blockchain status can be viewed by adding the contract address to the smart economy website code and a connection to the local testnet via MetaMask. By adding the private keys of the test environment, further transactions like paying corporate tax are executed and tested.

All bills

Show 10 entries Search:

billID	billValue	billSalesTax	billSum	billDate	billPaid	billOwner	billReceiver
53	1ether	20%	1.2ether	8/5/2021 - 11:39:36 AM	true	0xa55e0b6b86902c1c1ab159d6bbb6747a63a83aec	0x546074d12caa901eca800668f37ecfa21693c500
73	1ether	20%	1.2ether	9/28/2021 - 6:26:16 PM	true	0xa55e0b6b86902c1c1ab159d6bbb6747a63a83aec	0x546074d12caa901eca800668f37ecfa21693c500
10	1ether	20%	1.2ether	4/19/2021 - 5:32:56 PM	true	0x8cc2e76e8af3e46399c5f26aebbd839f4486a345	0x5db203a513484d369847d18b61c131ffa900d95b
30	1ether	20%	1.2ether	6/13/2021 - 7:46:16 PM	true	0x8cc2e76e8af3e46399c5f26aebbd839f4486a345	0x5db203a513484d369847d18b61c131ffa900d95b
50	1ether	20%	1.2ether	8/5/2021 - 11:39:36 AM	true	0x8cc2e76e8af3e46399c5f26aebbd839f4486a345	0x5db203a513484d369847d18b61c131ffa900d95b
70	1ether	20%	1.2ether	9/28/2021 - 6:26:16 PM	true	0x8cc2e76e8af3e46399c5f26aebbd839f4486a345	0x5db203a513484d369847d18b61c131ffa900d95b
86	1ether	20%	1.2ether	7/30/2022 - 9:32:56 AM	true	0x8ee1b7a5593aac362418f77f95a7a5504ed2eaa2	0x70a1e1d2ce14af803b89c44872dd21efab3dbf77
16	1ether	20%	1.2ether	5/9/2021 - 11:39:36 PM	true	0x5a66fe68ae9390118737c72f2234c60cb25b317b	0x8ee1b7a5593aac362418f77f95a7a5504ed2eaa2
36	1ether	20%	1.2ether	6/28/2021 - 9:46:16 AM	true	0x5a66fe68ae9390118737c72f2234c60cb25b317b	0x8ee1b7a5593aac362418f77f95a7a5504ed2eaa2
56	1ether	20%	1.2ether	8/24/2021 - 10:19:36 PM	true	0x5a66fe68ae9390118737c72f2234c60cb25b317b	0x8ee1b7a5593aac362418f77f95a7a5504ed2eaa2
billID	billValue	billSalesTax	billSum	billDate	billPaid	billOwner	billReceiver

Showing 31 to 40 of 87 entries Previous 1 2 3 4 5 ... 9 Next

Figure 6.9: Multiple year test results

All corporate income years

Show 10 entries Search:

corporateIncomeID	corporateIncomeYear	corporateIncomeSum	corporateTaxSum	corporateTaxPaid	payCorporateTax
12	2021	8 ether	2 ether	true	Pay corporate tax
23	2022	-1 ether	0 ether	false	Pay corporate tax
24	2023	-1.2423 ether	0 ether	false	Pay corporate tax
corporateIncomeID	corporateIncomeYear	corporateIncomeSum	corporateTaxSum	corporateTaxPaid	payCorporateTax

Showing 1 to 3 of 3 entries Previous 1 Next

Figure 6.10: Multiple year test results

All social insurance months

Show 10 entries Search:

socialInsuranceMonthID	socialInsuranceYear	socialInsuranceMonth	socialInsuranceSum	socialInsuranceAddress
1	2021	4	0.1 ether	0x1dacf12382cdd15dd2c289bf55c0d0f5c20688e9
6	2021	6	0.1 ether	0x1dacf12382cdd15dd2c289bf55c0d0f5c20688e9
11	2021	7	0.1 ether	0x1dacf12382cdd15dd2c289bf55c0d0f5c20688e9
16	2021	9	0.1 ether	0x1dacf12382cdd15dd2c289bf55c0d0f5c20688e9
3	2021	4	0.1 ether	0x416a081772abc4efbb80bd9ddee14604b514bf79
7	2021	6	0.1 ether	0x416a081772abc4efbb80bd9ddee14604b514bf79
13	2021	7	0.1 ether	0x416a081772abc4efbb80bd9ddee14604b514bf79
17	2021	9	0.1 ether	0x416a081772abc4efbb80bd9ddee14604b514bf79
21	2023	2	1 ether	0x416a081772abc4efbb80bd9ddee14604b514bf79
2	2021	4	0.1 ether	0x437f3dd7cd79733b89f661bfe7b71e95588291b6
socialInsuranceMonthID	socialInsuranceYear	socialInsuranceMonth	socialInsuranceSum	socialInsuranceAddress

Showing 1 to 10 of 22 entries Previous 1 2 3 Next

Figure 6.11: Multiple year test results

All income years

Show 10 entries

Search:

incomeID	incomeYear	incomeSum	incomeAddress
0	2021	3.2752 ether	0x9deeb245ea9cb8b9b53e17d126d536f0ce15e297
1	2021	0.32752 ether	0x1dacf12382cdd15dd2c289bf55c0d0f5c20688e9
2	2021	0.32752 ether	0x437f3dd7cd79733b89f661bfe7b71e95588291b6
3	2021	0.32752 ether	0x416a081772abc4efbb80bd9ddee14604b514bf79
4	2021	0.32752 ether	0xd4f7b33583a526551d611ec39781793824257c64
5	2021	0.32752 ether	0x546074d12caa901eca800668f37ecfa21693c500
6	2023	0.8188 ether	0x416a081772abc4efbb80bd9ddee14604b514bf79
incomeID	incomeYear	incomeSum	incomeAddress

Showing 1 to 7 of 7 entries

Previous 1 Next

Figure 6.12: Multiple year test results

CHAPTER 7

Analysis

In this part, the technical aspects of the implementation are analyzed with the findings of the tests. First, the functionality of the smart contract implementation is analyzed, before the two research questions of the testing part are examined.

- Is the current Ethereum blockchain able to carry out a sufficient number of transactions needed by the Austrian economy per year?
- Which modifications are necessary to test multiple year usage of the smart contract on a local network?
- Are smart contracts suitable to implement tax laws?

7.1 Functionality

The functionality of the smart contract behaves as expected with the performed tests. The report of the basic function test shows how the cash flow is redirected to the correct addresses and how the tax calculation is undertaken. The monthly social insurance calculation base and the yearly income of individual person addresses are stored correctly in the blockchain and can be viewed on the website. Business addresses are able to create bills, pay wages and pay their yearly corporate tax as expected. The yearly corporate income is also stored on the blockchain and can be viewed. If a bill receiver pays the bill, the price is calculated correctly for individual person addresses with VAT or without VAT for business addresses. The tax office address, local tax office and social insurance addresses are able to view and withdraw the collected taxes from the smart contract. This closes the part of the business cycle, which was the goal of this thesis. The smart contract shows that it is possible to implement a smart economy platform with an integrated tax system. The next step of testing would be with multiple test users and more optimization steps of the smart contract, although this is not possible with the resources of this master

thesis and thus is left to further research. Although the functionality is an important aspect of this research, this does not mean that the blockchain is capable of processing a sufficient number of transactions for an entire country.

7.2 Live performance estimation

For the live performance estimation, the gas usage of our test environment is used to calculate the number of transactions per block. Due to the low amount of transaction numbers compared with the other functions, contract creation, withdraw tax and pay corporate tax are not taken into account. The current gas limit is 12,500,000. For the pay wage function, the expensive gas price is used because the smart contract creates an object each month, which means that the transaction in each month will have a higher gas price. For the other functions, an average value between the higher and the lower gas amount is used.

Transaction Blocks per year	Gas usage Transactions per year	Transactions per block Transactions in Austria
Pay wage	480,000	26
2,339,650	**60,830,900**	36,530.712
Create bill	170,000	73
2,339,650	**170,794,450**	9,760,000,000*
Pay bill	81,600	153
2,339,650	**357,966,450**	9,760,000,000*
Pay bill business	130,000	96
2,339,650	**224,606,400**	9,760,000,000*

Table 7.1: Live performance estimation

*Only cashless transactions 1,952,000,000.

The calculation shows that the blockchain would be capable of processing the wage transaction, but it would need more than the half of the total transaction power to do so. Comparing the numbers of sale transactions, the weakness of the current live network is clearly visible. The current Ethereum implementation has a scalability problem and is not ready for mass adoption. The relatively low number of feasible transactions already leads to high gas prices, which means expensive transactions for every user. The miners always process transactions with a higher gas price first. A high demand leads to an increase of gas price because users have to bid a higher price to receive faster transactions. It does not slow down the blockchain as a whole given that the gas limit is separated from the gas price. Such expensive transactions are a major problem for the network because it makes it less usable for casual users and more complex smart contracts with high gas usage. In summary, the current Ethereum blockchain is unable to carry out a sufficient number of transactions needed by the Austrian economy per year.

7.3 Long-term test analysis

The modification of the elapsed time since contract creation works well by creating a new function that returns a later date. The modified contract has an adjustable time multiplier for the time since contract creation. In this master thesis, this is used to test contract functions that are only accessible after a year has passed, but with more qualitative data from the government and enhancements of the smart contract code for a specific problem statement it also could be used as a tool to simulate economy effects or analyze tax law changes. The number of transactions is still restricted by the capabilities of the testing environment, although the time between the transaction can be adjusted freely. Because the smart contract works with the current block time and some functions depending on this time, it is the only possibility to show the full capabilities of the smart contract in a reasonable amount of time. In summary, the analysis shows that theoretically a smart economy system could work but practically the network is not capable of processing such a high number of transactions.

7.4 Smart contracts and tax law

From the view of programming, the smart contract programming language solidity provides the functionality to execute automated tax calculation and transfer. A smart contract is able to comprise the main transactions of the Austrian tax system. Despite this, a solution for better performance and automated external input is needed. The entire tax system of a single country seems to complex for implementing on a global permissionless blockchain, mainly because the largest part of tax transactions are located in Austria and there is no need for publishing globally. Therefore a permissioned blockchain for local purposes would make sense.

CHAPTER 8

Conclusion and outlook

First of all, it is important to mention that the real taxation system in Austria is more complex than the implementation. Given the diverse business entities in an economy, there are exceptions for specific industries. Furthermore, the thesis completely ignores imports and exports that have their own tax laws. The focus of this thesis is not to implement the total complexity of the tax law, but rather it shows that it is technically possible to implement a simpler version of it. For developing the total complexity of the current law, more specific entities and multiple smart contracts are necessary. Based on this concept, the development of more specific tax functions is possible because the basic functionality for all relevant transaction is already implemented and works well. The implementation simplifies the economic processes to a level that includes all transactions. Therefore, the VAT percentage has to be inserted by the business address owner when creating a bill. This input needs knowledge about the tax law, whereby a business address owner needs an accountant for this purpose. In a more complex version, this input should also be automatically selected to reduce this source of errors. In my opinion, the current tax law is not suitable to be processed automatically. If someday taxes will become an automated process, there may be a shift how thinking about tax calculation from a legal prospect to a more logical, code-like approach. Despite the incapability with specialties of the tax laws, the created business cycle model from the beginning of the thesis is fully implemented and works flawlessly. Due to the current byte limit of a single contract, it is necessary to divide the code to multiple smart contracts and implement interfaces to connect them. Because the implementation is close to this byte limit, this is necessary for every extension of the code. Thinking bigger, each European country could have such smart contracts with interfaces for other countries to simplify the taxation across country borders.

The truffle testing environment with visual studio code is a suitable way to test the contract by creating automated transaction calls. The tests of the thesis can prove the functionality, but by including an economical database an accurate simulation of the

cash flow is possible. Even with the simple test data, it is interesting how the money is distributed over time. Besides the data, the income tax function would need to be revised for testing with real-world data. Due to the differential prices for higher income, this function is connected to the current price of Ethereum, which is continuously changing. To erase the fluctuation of the Ethereum price completely out of the picture of this smart contract, a ERC20 stable coin could be implemented. Essentially with some additions, the contract could also work with a digital Euro when invented. At present, there is an ERC20 token called DAI, which is implemented to hold the price of 1$ exactly for the purpose of having the possibility of a crypto currency with a stable price.

The current Ethereum blockchain is unable to process this number of transactions and even if it could, the gas prices for the transactions would be too high for mass usage like this. The capabilities of the chain have to increase drastically to be able to provide the transaction volume for mass products like a smart economy system. This brings us to the outlook of the Ethereum development. There are two approaches to scale the transaction volume. On the one hand, there are second-layer solutions in work to improve the transaction handling of an application for example zero knowledge proof. On the other hand, the Ethereum network is also still in development. At present, an ETH2 chain is developed to move from proof-of-work to a PoS system and implement sharding. Sharding splits the system to multiple synchronized chains and increases the scalability of the chain. If this project is successful, smart contracts for mass projects could be possible.

More effective and easier to handle is a permissioned sidechain to handle the amount of transactions and create interfaces to the main network. With such interfaces external transactions are possible and internal transactions are cheaper. Because of the locality of tax laws it is not necessary to implement on a global network, but an interface could simplify tax charge for foreign trades.

Nonetheless, it is necessary to set legal regulations to crypto currencies or crypto assets before it is used by a majority to protect the people from fraudulent projects and enable other projects to earn trust. In other industries, this is achieved through the introduction of standards and seals of approval. Global quality guidelines would open up the market for more reasonable and trustworthy companies and developers across the globe. In summary, there is still a long way to go to have a decentralized economy system, but the first steps have been taken. This thesis shows that it is possible to implement a system, but it also shows that the current state of development is not sufficiently advanced to run it on the Ethereum live chain.

List of Figures

List of Tables

Bibliography

[AACD20] Omar Ali, Mustafa Ally, Clutterbuck, and Yogesh Dwivedi. The state of play of blockchain technology in the financial services sector: A systematic literature review. *International Journal of Information Management; Volume 54, October 2020, 102199*, 2020.

[Anw19] Datta Anwitaman. Blockchain in the government technology fabric. *IIAS-Lien 2019 conference: Science, Technology and Innovation Policies track*, 2019.

[ARAL20] Afiya Ayman, Shanto Roy, Amin Alipour, and Aron Laszka. Smart contract development from the perspective of developers: Topics and issues discussed on social media. *Accepted for publication in the proceedings of the 4th Workshop on Trusted Smart Contracts (WTSC) in association with Financial Cryptography (FC 2020).*, 2020.

[AS19] Monika Di Angelo and Gernot Salzer. A survey of tools for analyzing ethereum smart contracts. *2019 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPCON)*, 2019.

[Ben20] Abdeljalil Beniiche. A study of blockchain oracles. *https://arxiv.org/abs/2004.07140*, 2020.

[BI19] Michael T. Belongia and Peter N. Ireland. A classical view of the business cycle. *Working Paper Series*, 2019.

[BKB+07] Pearl Brereton, Barbara A. Kitchenham, David Budgen, Mark Turner, and Mohamed Khalil. Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software*, 80:571–583, 2007.

[Blo] Blockgeeks. Smart contract platforms [a deep dive investigation]. `https://blockgeeks.com/guides/different-smart-contract-platforms`.

[CMO18] Giovanni Ciatto, Stefano Mariani, and Andrea Omicini. Blockchain for trustworthy coordination: A first study with linda and ethereum. *IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, 2018.

[con] consensys. Defi. https://consensys.net/blockchain-use-cases/decentralized-finance.

[eip] Eip-3675: Upgrade consensus to proof-of-stake. https://github.com/ethereum/EIPs/pull/3675.

[eth] ethereum.org. https://www.ethereum.org.

[Fin] Bundesministerium Finanzen. Steuern von a-z. https://www.bmf.gv.at/themen/steuern/steuern-von-a-bis-z.html.

[FMHC20] Md Sadek Ferdous, Jabed Mohammad Morshed, Mohammad A. Hoque, and Alan Coleman. Blockchain consensus algorithms: A survey. *arXiv:2001.07091*, 2020.

[gol] golden. https://golden.com/wiki/Metamask.

[HMPR04] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *MIS Quarterly*, 28:75–105, 2004.

[JSK+17] Aljosha Judmayer, Nicholas Stifter, Katharina Krombholz, Edgar Weippl, Elisa Bertino, and Ravi Sandhu. Blocks and chains: Introduction to bitcoin, cryptocurrencies, and their consensus mechanisms. *Synthesis Lectures on Information Security, Privacy, & Trust*, 9(1):1–123, 2017.

[LXB+21] Qinghua Lu, Xiwei Xu, H.M.N. Dilum Bandara, Shiping Chen, and Liming Zhu. Patterns for blockchain-based payment applications. *arXiv:2102.09810v2*, 2021.

[met] Metamask. https://metamask.io.

[min] Mining. https://ethereum.org/en/developers/docs/consensus-mechanisms/pow/mining/.

[Moh] Martin Mohr. Nur bares ist wahres. https://de.statista.com/infografik/8808/anteile-von-zahlungsmitteln-in-oesterreich.

[ora] Oracles. https://ethereum.org/en/developers/docs/oracles/.

[PB17] Joseph Poon and Vitalik Buterin. Plasma: Scalable autonomous smart contracts. *https://plasma.io/*, 2017.

[PP15] Gareth W. Peters and Efstathios Panayi. Understanding modern banking ledgers through blockchain technologies: Future of transaction processing and smart contracts on the internet of money. *Available at SSRN: https://ssrn.com/abstract=26 or http://dx.doi.org/10.2139/ssrn.2692487*, 2015.

[PP19] Gareth W. Peters and Efstathios Panayi. An empirical study into the success of listed smart contracts in ethereum. *IEEE Access. PP. 1-1. 10.1109/ACCESS.2019.2957284.*, 2019.

[Rob19] Peter Robinson. The merits of using ethereum mainnet as a coordination blockchain for ethereum private sidechains. *Published by Cambridge University Press*, 2019.

[SLL17] Karol Szomolányi, Martin Lukáčik, and Adriana Lukáčiková. Business cycles in european post-communist countries. *Contemporary Economics*, 11(2):171–186, 2017.

[SLS20] Chiara Spadafora, Riccardo Longo, and Massimilano Sala. A coercion-resistant blockchain-based e-voting protocol with receipts. *Advances in Mathematics of Communications, doi: 10.3934/amc.2021005*, 2020.

[Soe21] Mark Soelman. Permissioned blockchains: A comparative study. *University of Groningen, Faculty of Science and Engineering*, 2021.

[sol] solidity.readthedocs.io. Solidity documentary. `https://solidity.readthedocs.io/en/v0.5.12`.

[staa] statista. `https://de.statista.com/statistik/daten/studie/202813/umfrage/eu-laender-mit-den-meisten-transaktionen-im-bargeldlosen-zahlungsverkeh`

[Stab] Bootstrap 4 Get Started. w3schools. `https://www.w3schools.com/bootstrap4/bootstrap_get_started.asp`.

[WE20] Martin Westerkamp and Jacob Eberhardt. zkrelay: Facilitating sidechains using zksnark-based chain-relays. *IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, 2020.

[wika] wikibooks.org. Wirtschaftsteilnehmer. `https://de.wikibooks.org/wiki/Betriebswirtschaft/_Grundlagen/_Wirtschaftsteilnehmer`.

[wikb] wikipedia.org. Wirtschaftskreislauf. `https://de.wikipedia.org/wiki/Wirtschaftskreislauf`.

[YLS+18] Bin Yu, Joseph Liu, Amin Sakzad, Surya Nepal, Ron Steinfeld, Paul Rimba, and Man Ho Au. Platform-independent secure blockchain-based. *Chen L., Manulis M., Schneider S. (eds) Information Security. ISC 2018. Lecture Notes in Computer Science, vol 11060. Springer, Cham.*, 2018.

[YMRP19] Renlord Yang, Toby Murray, Paul Rimba, and Udaya Parampalli. Empirically analyzing ethereum's gas mechanism. *4TH IEEE EUROPEAN SYMPOSIUM ON SECURITY AND PRIVACY WORKSHOPS (EUROS&PW)*, 2019.

[YW18] Yong Yuan and Fei-Yue Wang. Blockchain and cryptocurrencies: Model, techniques, and applications. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 48:1421–1428, 2018.

Attachment

Review Protocol - Smart economy

Research questions

How can state of the art blockchain systems help us to automate economic processes?

Model the participants and functions of Austrian economy. How can we define a business cycle between governments, social insurance, companies and consumer?

A systematic review

Organization, City, Country: TU Wien, Wien, Austria

Prepared by: Ing. Oliver Steizinger BSc

Date: February 2020

Supervisor: Privatdoz. Mag.rer.soc.oec. Dipl.-Ing. Dr.techn. Weippl Edgar

Research team members: Ing. Oliver Steizinger BSc

Table of Contents

1.0 Background

Business informatics at TU-Wien is a study of informatics, economy and the connection between these disciplines. From the perspective of a computer scientist, the models which are used to calculate and predict the economy seem simple in the first place but can evolve to complex structures and are always influenced by social phenomena. That's why you have to assume a lot of parameters and use simplifying solutions. It's not a natural science like physics or biology. Economics is man-made and doesn't follow natural laws and is not entirely consequential like other sciences. As a programmer you have to accept that the current economic system is not entirely computable or predictable. Social issues have impact on the economic system what makes it impossible to compute exactly what will happen. Simulations are thus an important alternative for studying our economic model. This brings us to the motivation for this research: I want to create a small economic system within the decentralized Ethereum blockchain platform and build a test environment to find out the limitations of the current Ethereum implementation. This simulation should demonstrate how an automated economic system could work.

2.0 Objective

1. To investigate the current state of the art blockchain systems and their relevance in future economic processes.

2. To create a business cycle model for the Austrian economy including all business participants and business functions within the country.

3.0 Review Questions

For the purposes of this literature review, the population, intervention, comparators and outcomes (PICO) framework to inform the review objectives are presented below.

3.1 Blockchain systems

Population	Intervention	Comparison	Outcome
Programmable blockchain systems (smart contracts)	Ability to develop business functions within the blockchain system	Unability to develop business functions within the blockchain system	Quality and relevance of running blockchain systems

3.2 Business Cycle

Population	Intervention	Comparison	Outcome
Business cycle models	Relevance for a business cycle model in Austria	No relevance for a Business Cycle model in Austria	Gather relevant information for development of a business cycle model in Austria

4.0 Evidence gathering and study selection

See **Appendix A** for the list of databases, websites and journals which may be searched.

4.1 Evidence gathering

The evidence gathering approach will have four components:

4.1.1 Searching databases

The databases in the table below will be searched with a pre-determined strategy as detailed in **Appendix B**. Due to lack of resources and manpower a small number of databases available from the TU Wien library will be used for the literature research.

Topic/Field	Database
Computer science	IEEE Xplore / Electronic Library Online (IEL)
Economics	ABI/INFORM Collection
Interdisciplinary	Scopus

4.1.2 Hand searching

The following websites and whitepapers will be hand-searched for relevant information:

Resources that will be searched by hand
Websites: • https://www.blockchainforscience.com/ • https://www.frontiersin.org/journals/blockchain • https://www.blockchainstudies.org/ • http://blockchain.mit.edu/ • https://www.blockchain-lab.org/ • http://blockchain.cs.ucl.ac.uk/ • https://www.blockchainresearchinstitute.org/ Whitepapers: • Ethereum • RSK • Stellar • TRON • NEO

4.1.3 Reference searches

Bibliographies of those papers that match the eligibility criteria below will be searched by hand to identify any further, relevant references, which will be subject to the same screening and selection process.

4.2 Eligibility criteria

After gathering the evidence, the following eligibility criteria will be applied to the results and all identified references screened using a three-stage approach to reviewing the title, abstract and conclusion. The list of relevant research after reviewing the title is attached in **Appendix C**.

4.2.1 Types of studies

All types of evaluative study designs are eligible for inclusion, including grey literature. Studies will not be selected on methodological quality. Furthermore the whitepapers of selected blockchain systems will be reviewed.

4.2.2 Types of systems

Blockchain systems:

All programmable blockchain systems will be reviewed. In other words, the reviewed systems must have the ability to write smart contracts. The research will include a list of advantages and disadvantages of each relevant blockchain platform. Moreover blockchains with doubtful decentralized algorithms will be rejected. Systems with a life cycle of at least two years will be examined. Furthermore literatures similar to the research topic are examined.

Business cycles:

The main target of this review is to gather a theoretical background of business cycle systems. Therefore basic literature about business cycles will be reviewed. Furthermore business models of European countries with similarity to the Austrian economy will be examined.

4.2.3 Outcome

Blockchain systems:

The outcome of the blockchain-literature review is a general description of smart contract systems and a comparison of relevant smart contract systems. This information is used for the selection of the smart contract system in this thesis.

Business cycles:

The business cycle research will be performed to build a model of the Austrian economy model. Therefore the outcome of the literature review will be a basic theoretical analysis of business cycles. Furthermore existing business models will be reviewed.

4.3 Exclusion criteria

Due to the fact that blockchain systems are a new technology, papers older than five years will be excluded. Business cycle literature older than ten years will be excluded due to lack of resources of this project. Failure to meet any one of the above eligibility criteria (section 4.2) will result in exclusion from the review. The number of excluded studies (including reasons for exclusion for those excluded following review of the full text) will be recorded at each stage.

5.0 Data extraction

Data will be extracted from relevant papers using predefined information specification templates attached in **Appendix c**. Data will be collected regarding the reasons for exclusion, characteristics of included studies, systems and outcome.

Appendix A

Databases

Databases for computer science:

- ACM: Association for Computing Machinery Digital Library
- IEEE Xplore (IEE/IEEE)
- Scopus (Elsevier)
- Inspec: Engineering Research Database
- Web of Science (Clarivate Analytics)
- ProQuest
- CiteSeerX
- DBLP
- JSTOR
- Compendex
- The Collection of Computer Science Bibliographies
- ZDE Elektrotechnik, Elektronik und Informationstechnik

Databases for economics:

- ABI/INFORM Collection
- AgEcon Search
- BEFO Betriebsführung und Betriebsorganisation
- EconBiz
- EconLit
- NBER: National Bureau of Economic Research
- RePEc: Research Papers in Economics
- Social Sciences Citation Index (Web of Science)
- OECD iLibrary
- International Historical Statistics
- JSTOR
- Scopus

General databases:

- Google Scholar

e-Journal Collections

Databases for computer science:

- SpringerLink
- ScienceDirect
- Wiley Online Library

Resources for search by hand
Websites: • https://www.blockchainforscience.com/ • https://www.frontiersin.org/journals/blockchain • https://www.blockchainstudies.org/ • http://blockchain.mit.edu/ • https://www.blockchain-lab.org/ • http://blockchain.cs.ucl.ac.uk/ • https://www.blockchainresearchinstitute.org/ Whitepapers: • Ethereum • EOS • Cardano • RSK • Stellar • TRON • Hyperledger • Parity • NEO

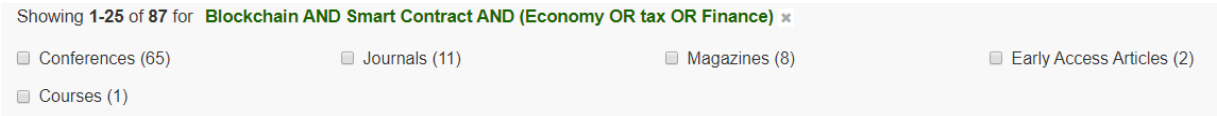
Appendix B

Blockchain search strategy:

Blockchain AND Smart Contract AND (Economy OR tax OR Finance)

Blockchain search results:

IEEE



Scopus

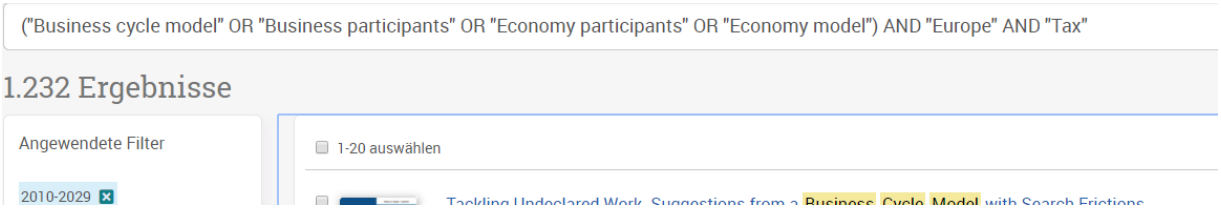
Business Cycle search strategy:

("Business cycle model" OR "Business participants" OR "Economy participants" OR "Economy model") AND "Europe" AND "Tax"

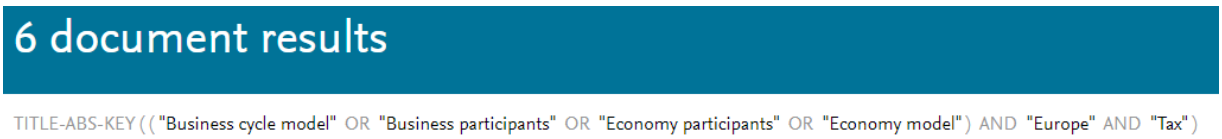
Publication dates from 2010 to now

Business Cycle search results:

ABI/INFORM



Scopus



Appendix C – Paper Reviews

Paper Reviews - Blockchain

Searching databases

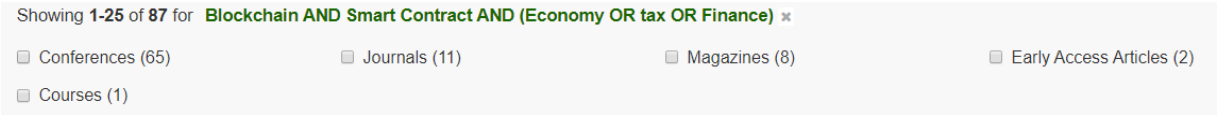
Topic/Field	Database
Computer science	IEEE Xplore / Electronic Library Online (IEL)
Interdisciplinary	Scopus

Search query

Blockchain AND Smart Contract AND (Economy OR tax OR Finance)

Search results

IEEE



Scopus

Organization, City, Country: TU Wien, Wien, Austria

Prepared by: Ing. Oliver Steizinger BSc

Date: February 2020

Research team members: Ing. Oliver Steizinger BSc

Title: Blockchain-Enabled Smart Contracts: Architecture, Applications, and Future Trends

Year/Source: 2019/IEEE

Author: Shuai Wang ; Liwei Ouyang ; Yong Yuan ; Xiaochun Ni ; Xuan Han ; Fei-Yue Wang

Abstract Summary:

The first part of the abstract is general information about blockchain and a reminder of the development status of such systems. Then the three parts of the paper are described.

- Introduction of the operating mechanism and mainstream platforms of blockchain-enabled smart contracts, and proposed a research framework for smart contracts based on a novel six-layer architecture
- Both the technical and legal challenges, as well as the recent research progresses, are listed
- Presentation of several typical application scenarios
- Discussion of the future development trends of smart contracts

Conclusion Summary:

In the conclusion the authors bring out the popularity of blockchain and smart contracts in both academic and industrial communities. Smart contracts are expected to revolutionize traditional industries without involvements of a trusted authority or a central server. Furthermore the conclusion describes the four parts of the paper like in the abstract.

Title: Towards a Decentralized Data Marketplace for Smart Cities

Year/Source: 2018/IEEE

Author: Gowri Sankar Ramachandran ; Rahul Radhakrishnan ; Bhaskar Krishnamachari

Abstract Summary:

Due to the fact that prior projects have examined centralized data marketplaces for smart cities, this paper explores how a decentralized data marketplace could be created using blockchain and other distributed ledger technologies. It considers possible benefits, identifies different elements and shows how these elements could be potentially integrated into a solution. Furthermore a simple smart contract of a decentralized registry is presented.

Conclusion Summary:

In the conclusion following challenges of decentralized marketplaces are presented:

- Managing system complexity
- Economic incentives and centralization
- Applications and interfaces

In ongoing work the researcher will try to refine the various elements of the architecture and build a working prototype that assembles all the pieces into a functioning system.

Title: Analyzing Financial Smart Contracts for Blockchain

Year/Source: 2018/IEEE

Author: Muskan Vinayak ; Har Amrit Pal Singh Panesar ; Saulo dos Santos ; Ruppa K. Thulasiram ; Parimala Thulasiraman ; S.S. Appadoo

Abstract Summary:

The first part of the abstract mentions that crypto currencies evolved to smart contracts which can be used for different applications such as Option Pricing, Currency Exchange, Revenue Management System, Crowd-funding and Peer-to-Peer networking. The aim of the paper is to construct a smart contract that could be used to take various possible positions in a European style option. Potential security vulnerabilities are analyzed.

Conclusion Summary:

Two main contributions of the study:

- Creating a Smart Contract for depicting the functionality of an European Option
- Analyzing the developed contract using a pre-existing tool Oyente

Possible scenarios:

- user sending sufficient premium or not for holding/buying a call option. Also, if the contract has sufficient balance to be fixed for settlement
- user sending enough money to be fixed for settlement of the contract or not being a writer of the call option

Some of the essential conditions required to properly implement the contract on Blockchain are not currently available and hence are depicted as flaws in Oyente.

Title: Development of a Road Tax Payment Application using the Ethereum Platform

Year/Source: 2018/IEEE

Author: Daniel Zinca ; Vlad-Andrei Negrean

Abstract:

This paper describes the development of a road tax payment application using the Ethereum platform. The web solution was developed using the ReactJS platform. The smart contract was developed using Remix.

Conclusion:

The proposed solution describes a system for the online payment of the road tax that uses cryptocurrencies and blockchain for increased database security and faster international transaction processing.

The system is comprised of two parts: the Ethereum contract and the web application.

The solution stands as a viable one for the road tax payment due to lower costs, processing speed and increased security.

Extended features that we intend to implement in the foreseeable future:

- Multiple tax pay (not limited to driving related activities) including driving penalties, household taxes, etc.
- Alternative cryptocurrency payment options (Bitcoin, Litecoin, etc).
- Adjusting the application interface for mobile platforms.
- API development for instant checking by the traffic surveillance solutions.

Title: Blockchain for Trustworthy Coordination: A First Study with LINDA and Ethereum

Year/Source: 2018/IEEE

Author: Giovanni Ciatto ; Stefano Mariani ; Andrea Omicini

Abstract Summary:

The study is focused on the Ethereum blockchain technology, map it onto LINDA tuple-based coordination model, and discuss two proof-of-concept implementations of LINDA on Ethereum. Conceptual and technical feasibility of blockchain-based coordination in multi-agent systems are discussed in the study.

Conclusion Summary:

Goal:

- reporting about feasibility of implementing a LINDA-like tuple-based coordination service on top of a reference blockchain technology
- shedding some light on the most notable issues arising when doing so—e.g., the economical impact of performing coordination operations

The paper shows how LINDA can be implemented on Ethereum and issues are analyzed.

Next steps:

- performing a comparison of different implementations of our contract space concept on top of different blockchain technologies, thus of different smart contracts implementations - for instance HLF and Corda
- defining a rigorous formalisation of the semantics behind different blockchain and smart contract models in terms of their potential coordination capabilities

Title: Blockchains as Enablers for Auditing Cooperative Circular Economy Networks

Year/Source: 2018/IEEE

Author: George Alexandris ; Vassilis Katos ; Sofia Alexaki ; George Hatzivasilis

Abstract Summary:

In this study a collaborative circular economy business model is proposed, where the circular economy cycle is materialized by assets transitioning between asset operators on a demand driven approach. The common view of asset state between all parties can be enabled by blockchains and smart contracts, which can provide the underlying technology to share data with integrity, while simultaneously offering more efficient interoperability between participants. A conceptual asset record and sharing mechanism is presented.

Conclusion Summary:

Governmental environmental authorities by virtue of their role as a regulator can leverage the benefits of blockchains while retaining a sufficient degree of control over the blockchain application. This makes a common view of all monitored assets possible, shared by all current and prospective controlling entities of assets, while at the same time, and ensures that asset operators retain complete control of their asset's state record. Smart Contracts play a pivotal role towards offering granular and dynamic control of a state record. We have shown that they can be flexible enough to satisfy the main requirements for implementing and accessing asset records.

Title: What Is the Blockchain?

Year/Source: 2017/IEEE

Author: Massimo Di Pierro

Abstract:

Blockchain is a new technology, based on hashing, which is at the foundation of the platforms for trading cryptocurrencies and executing smart contracts. This article reviews the basic ideas of this technology and provides a sample minimalist implementation in Python.

Conclusion Summary:

The conclusion of the paper points out that there are different cryptocurrencies run on different platforms and make different storage and hashing choices. Furthermore there are different implementations of the algorithm for a single cryptocurrencies. The functionality of a smart contract is described briefly.

Title: Validation and Verification of Smart Contracts: A Research Agenda

Year/Source: 2017/IEEE

Author: Daniele Magazzeni ; Peter McBurney ; William Nash

Abstract:

Smart contracts might encode legal contracts written in natural language to represent the contracting parties' shared understandings and intentions. The issues and research challenges involved in the validation and verification of smart contracts, particularly those running over blockchains and distributed ledgers, are explored.

Summary:

This paper is a detailed technical description of blockchain and smart contracts.

Title: Smart-CPR: Self-Organisation and Self-Governance in the Sharing Economy

Year/Source: 2017/IEEE

Author: David Burth Kurka ; Jeremy Pitt

Abstract Summary:

The paper shows that it is possible to develop a system for common-pool resource management with smart contract technologies. The results demonstrate that the model -- the Smart-CPR - is able to distribute resources efficiently and is capable of detecting and punishing non-compliant or unhelpful behavior.

Conclusion:

In this work we demonstrated that with the automation brought with smart-contracts and blockchain, combined with the computational justice framework, it is possible to build a system to solve the problem of distributed supply and demand in efficient and self-organised ways.

We have shown how the Smart-CPR can efficiently act scheduling allocations in cases of full compliance and how processes of self-organisation in the rules' definition prevent the abuses of malicious agents.

Title: The Advantages and Disadvantages of the Blockchain Technology

Year/Source: 2018/IEEE

Author: Julija Golosova ; Andrejs Romanovs

Abstract Summary:

In this paper the description of the Blockchain technology, and it advantages and disadvantages are analyzed. Many already implemented applications of Blockchain technology were studied, as well as affected success or problems factors during the implementations.

Conclusion Summary:

Advantages and dangers of blockchain systems are briefly discussed.

Title: Blockchains and International Business

Year/Source: 2019/IEEE

Author: Nir Kshetri

Abstract:

Discusses how blockchain is transforming international business practices and relatioships. Blockchain and smart contracts are transforming international trade activities. Proof-of-concepts (PoCs), prototypes, pilot projects, and actual deployments indicate that smart contracts can bring benefits to those involved in trading.

Conclusion Summary:

In the summary some benefits of blockchain solutions in economy are discussed.

Title: Lowering Financial Inclusion Barriers with a Blockchain-Based Capital Transfer System

Year/Source: 2019/IEEE

Author: Alex Norta ; Benjamin Leiding ; Alexi Lane

Abstract Summary:

The abstract is a description how crypto currencies can improve financial transactions focused on the crypto currency Everex.

Conclusion Summary:

In the conclusion the benefits of a crypto-based capital transfer system with stable coins like Everex are discussed. A short preview of future releases with functionality for lending are presented.

Title: Design and Implementation of Financial Smart Contract Services on Blockchain

Year/Source: 2019/IEEE

Author: Vinayak, M., Santos, S.D., Thulasiram, R.K., Thulasiraman, P., Appadoo, S.S.

Abstract Summary:

In this paper, an implementation of the financial instrument, a collateral smart contract services (CSCS) with Hyperledger Fabric network is presented.

Conclusion Summary:

A financial smart contract for collateral services on Hyperledger Fabric has been successfully designed. Challenges for programming with Hyperledger Fabric are discussed.

Title: Smart contract programming languages on blockchains: An empirical evaluation of usability and security

Year/Source: 2018/SpringerLink

Author: Parizi, R.M., Amritraj, Dehghantanha, A.

Abstract Summary:

Because current research on contract development is not sufficient and is still in a stage of infancy, the paper gives a comprehensive analysis of domain-specific programming practices from critical points of usability and security.

Conclusion:

The given evaluation included an experiment that was performed to compare the usability and security vulnerability of the three domain-specific languages, namely Solidity, Pact and Liquidity. The experiment results demonstrated that although Solidity is the most usable language for a new developer to program smart contracts, it is the least secure language to vulnerabilities. On the other hand, Liquidity and Pact show lower usability but seem secure for now. Consequently, our results contribute to the body of experimental evidence about the usability and security of the smart contract programming languages, which is currently scarce.

Paper Reviews – Business Cycle

Searching databases

Topic/Field	Database
Economics	ABI/INFORM Collection
Interdisciplinary	Scopus

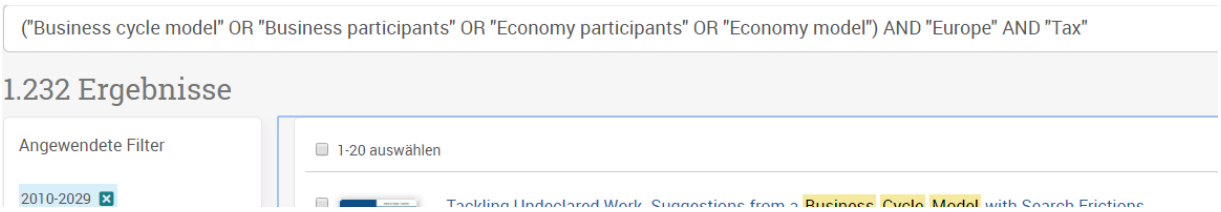
Search query

("Business cycle model" OR "Business participants" OR "Economy participants" OR "Economy model") AND "Europe" AND "Tax"

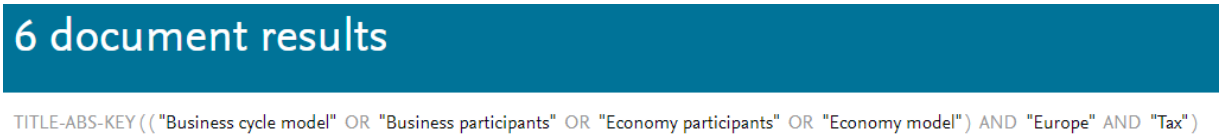
Publication dates from 2010 to now

Search results

ABI/INFORM



Scopus



Organization, City, Country: TU Wien, Wien, Austria

Prepared by: Ing. Oliver Steizinger BSc

Date: February 2020

Research team members: Ing. Oliver Steizinger BSc

Title: Generational policy and aging in closed and open dynamic general equilibrium models

Year/Source: 2013/Handbook of Computable General Equilibrium Modeling

Author: Fehr, H., Jokisch, S., Kallweit, M., Kindermann, F., Kotlikoff, L.J.

Abstract Summary:

The chapter examines the micro- and macroeconomic effects of generational policies using closed and open general equilibrium dynamic life-cycle models. It is a mathematical solution including country-specific tax, spending, social security, healthcare policy, deficit policy age-cohort- and country-specific mortality, age-specific fertility, age-specific morbidity, lifespan uncertainty, age- and skill-specific emigration and immigration, earnings inequality driven by skill differences and idiosyncratic labor earnings uncertainty, capital adjustment costs, international trade, international capital flows, trade specialization, and trade policy.

Title: A Tale of Tax Policies in Open Economies

Year/Source: 2011/IDEAS Working Paper Series from RePEc; St. Louis

Author: Auray, Stéphane; Eyquem, Aurélien; Gomme, Paul

Abstract Summary:

This paper develops an open economy model to evaluate the impact of various permanent tax changes. The model is calibrated to the U.S. and some different cases to analyze are discussed.

Conclusion Summary:

The results of the experiment are discussed.

Title: Optimal Fiscal Policy in the Presence of VAT Evasion: The Case of Bulgaria

Year/Source: 2018/Finance a Uver; Prague

Author: Vasilev, Aleksandar

Abstract Summary:

The paper uses a dynamic general-equilibrium model, calibrated to Bulgarian data (1999-2014) for computational experiment findings.

- The optimal steady-state income tax rate is zero
- The benevolent Ramsey planner provides the optimal amount of the valuable public services, which are now three times lower
- The size of the grey sector is twice lower
- optimal steady-state consumption tax needed to finance the optimal level of government spending is twice lower, as compared to the exogenous policy case

Conclusion Summary:

The conclusion is similar to the abstract.

Title: A MULTINATIONAL ANALYSIS OF TAX RATES AND ECONOMIC ACTIVITY

Year/Source: 2011/Journal of International Business Research, Beil. Special Issue; Arden

Author: Smith, Lawrence C, Jr; Smith, L Murphy; Gruben, William C

Abstract Summary:

The purpose of this study is to examine the relationship between tax rates in selected countries and economic activity, including GDP growth, unemployment, and savings. The sample of countries used in the study consists of the Organization of Economic Cooperation and Development (OECD) countries. Results indicate that lower tax rates are associated with more favorable economic activity, including growth in GDP, change in unemployment, and change in savings.

Conclusion Summary:

Results are discussed. Results are mixed but reveal some meaningful relationships between tax rates and economic activity.

Title: Tax optimization in an agent-based model of real-time spectrum secondary market

Year/Source: 2017/Telecommunication Systems; New York

Author: Gazda, Juraj; Kovác, Viliam; Tóth, Peter; Drotár, Peter; Gazda, Vladimír

Abstract Summary:

This paper aims to build an agent based model of the real-time secondary spectrum market in which various taxes including value-added tax, corporate tax, consumption tax and fixed tax, are employed. The results of the analysis confirm the existence of a tax distortion, i.e. a system deviation from the efficient system functioning affected by the tax introduction.

Conclusion Summary:

In this paper, following impacts on the functioning of the real-time secondary spectrum market are considered

- corporate tax
- consumption tax
- value-added tax
- fixed tax

Different methods and results are discussed.

Title: A Theory of Optimal Capital Taxation

Year/Source: 2012/NBER Working Paper Series; Cambridge

Author: Piketty, Thomas; Saez, Emmanuel

Abstract Summary:

This paper develops a realistic, tractable theoretical model that can be used to investigate socially-optimal capital taxation. The parameters of the dynamic model are discussed. Finally, its discussed how adding capital market imperfections and uninsurable shocks to rates of return to our optimal tax model leads to shifting one-off inheritance taxation toward lifetime capital taxation, and can account for the actual structure and mix of inheritance and capital taxation.

Conclusion Summary:

With the results of the model, assumptions of optimal capital taxation are presented.

Title: A low growth path in Austria: potential causes, consequences and policy options

Year/Source: 2014/Empirica; New York

Author: Stocker, Andrea; Großmann, Anett; Hinterberger, Friedrich; Wolter, Marc Ingo

Abstract Summary:

This paper reports on an Austrian research project that deals with the question how the Austrian society could cope with long-lasting low economic growth. The consequences of a low economic growth are discussed.

Conclusion Summary:

The results show that the macroeconomic consequences of low economic growth in Austria are substantial: the labour market suffers from a shortage of labor supply (due to reduced migration) and from a reduced demand for labor (due to reduced demand in consumption, investments and exports). The decrease in employment in the integration scenario leads to a negative development of the disposable income of private households (tax rates and social security contributions held constant). Compared to the reference scenario, public debt is higher. Due to the assumption of slight population growth, public expenditures grow slower, however, tax incomes decrease at a higher rate.

To analyze whether and how policy measures are able to cope with the negative consequences of persistent low growth, four measures were chosen.

- a cost-neutral reduction of working time by 10 %
- the introduction of a cost-neutral eco-social reform of levies
- a reduction of environmentally harmful subsidies
- the promotion of the private demand for services

The selected policy measures are suitable to reduce the negative economic effects of low growth.

Title: Business Cycles in European Post-Communist Countries

Year/Source: 2017/Contemporary Economics; Warsaw

Author: Szomolányi, Karol; Lukáčik, Martin; Lukáčiková, Adriana

Abstract Summary:

This is a business cycle study of chosen European post-communist countries. Economic activity in the studied countries is relatively low and volatile, and the trade balance and government purchases have a relatively significant countercyclical character. In the study, both traditional and contemporary business cycle definitions are used.

Conclusion Summary:

Business cycle characteristics differ slightly from those of emerging countries around the world:

- post-communist countries' expansions and recessions are longer
- post-communist countries' recessions are more pronounced
- post-communist countries' output is relatively low in volatility
- post-communist countries' household consumption is relatively highly volatile
- the government share of GDP is countercyclical in post-communist countries
- trade balances (current accounts) are relatively strongly counter-cyclical in post-communist countries

Title: Political business cycles 40 years after Nordhaus

Year/Source: 2016/Public Choice; Dordrecht

Author: Dubois, Eric

Abstract:

The aim of this article is to survey the huge literature that has emerged in the last four decades following Nordhaus's (Rev Econ Stud 42(2):169–190, 1975) publication on political business cycles (PBCs). I first propose some developments in history of thought to examine the context in which this ground-breaking contribution saw the light of the day. I also present a simplified version of Nordhaus's model to highlight his key results. I detail some early critiques of this model and the fields of investigations to which they gave birth. I then focus on the institutional context and examine its influence on PBCs, the actual research agenda. Finally, I derive some paths for future research.

Conclusion Summary:

The conclusion is a outlook for future research.

Title: Model of the circular economy and its application in business practice

Year/Source: 2019/Environment, Development and Sustainability; Dordrecht

Author: Ungerman, Otakar; Dědková, Jaroslava

Abstract Summary:

The aim of this paper is to put together a model of the circular economy to determine the economic result of enterprises' involvement in the circular economy. The model was applied on the basis of primary research aimed at determining the level of enterprises' involvement in renovation, reuse and recycling, and hard data were acquired from the Czech Statistical Office. The results exactly proved that enterprises profit through their involvement in the circular economy. A loss was demonstrated in just one sector and one phase of processing discarded waste. The results demonstrate a positive economic impact in the long term. The results of this work clearly prove the claim that the circular economy has a positive impact on the environment, and also on enterprises' economic prosperity and thus the aggregate economy of the state.

Conclusion Summary:

In general, the main benefit of the circular economy may be summarised as a positive impact on environmental sustainability.

Results and methodology are discussed.

Title: A Classical View of the Business Cycle

Year/Source: 2019/NBER Working Paper Series; Cambridge

Author: Michael T. Belongia; Peter N. Ireland

Abstract Summary:

This paper develops a structural vector autoregressive time series model that allows these "classical" channels of monetary transmission to operate alongside the now-morefamiliar interest rate channel of the New Keynesian model. Even with Bayesian priors that intentionally favor the New Keynesian view, the United States data produce posterior distributions for the model's key parameters that are consistent with the ideas of Fisher and Working. Changes in real money balances enter importantly into the model's aggregate demand relationship, while growth in Divisia M2 appears in the estimated monetary policy rule.

Conclusion Summary:

Though based on New Keynesian priors, the expanded VAR estimated here provides a posterior view of monetary policy and its effects on the economy that is highly “classical” instead. Within this estimated model, changes in real money balances play a role, alongside movements in real interest rates, in transmitting the effects of monetary policy to aggregate output over the period during which nominal rigidities prevent prices from adjusting fully. Likewise, changes in nominal money growth signal, much more clearly than changes in nominal interest rates, whether monetary policy is expansionary or contractionary. The estimated model attributes sizable movements in inflation and the output gap to monetary policy disturbances, particularly during the disinflationary recessions of the early 1980s and the Great Recession of 2007-2009.

Results are compared to Fisher (1923, 1925, 1926) and Working (1923, 1926).