*Division of Computing Science and Mathematics*

*Faculty of Natural Sciences*

*University of Stirling*

How could a blockchain insurance application (Dapp) solve problems in traditional insurance?

Jelmer van Ginkel

**Dissertation submitted in partial fulfilment for the degree of   
Master of Science in Fintech**

**September 2019**

# Abstract

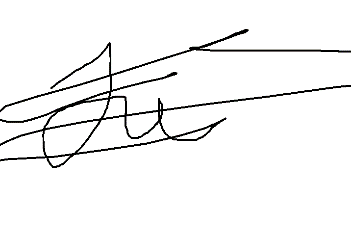
Insurance has become larger than ever, mainly due to the effects of risk diversification. Although, it seems that having a large centralized intermediary might not be the most efficient way to apply insurance. Blockchain technology has decentralization properties which can be used to create insurance products in a decentralized matter and disintermediate the current industry. Although, not every blockchain has the capability of performing such complex tasks. Ethereum does have these capabilities and is therefore used. A few use cases where considered to create a proof of concept. Earthquake insurance seemed the most applicable use-case due to availability of data and the potential to implement as a parametric insurance. Etherisc provides a smart contract which can function as basis to build new insurance products on top. This framework is called GIF(Generalized Insurance Framework). Besides Etherisc, Provable provides a similar framework to use oracle queries. Provable will be used to import data into the Ethereum blockchain. The application has a few general phases which are derived from Etherisc’s GIF. The earthquake insurance starts with an application and premium quote. To receive a premium quote a risk assessment will be made through a computation query. The query accesses a python script via IPFS which is run on a virtual machine. The result will be a RiskScore and is then imported back to the blockchain. The score is used to calculate the premium. When the premium is received an application can be submitted. The underwriter is able to decline or accept the application. When accepted the policy will be active and claims can be submitted. A claim can be submitted and verified by an oracle that checks the location and magnitude of the latest earthquake resulting in a positive or negative ClaimScore. If positive the claim will be verified and the pay-out amount is equal to the coverage. Before the pay-out a manual check will decide whether to accepted or rejected.

# Attestation

I understand the nature of plagiarism, and I am aware of the University’s policy on this.

I certify that this dissertation reports original work by me during my University project except for the following (*adjust the list below according to the circumstances*):

* The haversine formula in the Python scripts was largely taken from [1]
* The code of the Provable queries in appendix 8: RiskAssessment.sol and 9: ClaimVerification.sol was based of the examples in from [2].
* The code of the Docker files in appendix 4: Dockerfile Risk Assessment and 5: Dockerfile ClaimVerification was based of the examples in from [2].
* The code of the main smart contract in appendix 10: EarthquakeInsurance.sol and appendix 11: ImplementationEtherisc.sol were based of the tutorial in [3].

**Signature:**   **Date: 02-09-2019**

# Acknowledgements

First of all, I would like to thank my family and girlfriend for always supporting me in my journey in becoming the best version of myself. Learning new challenging skills, acquire broader perspective of the world and my pursuit of all the subjects I am interested in. Secondly, I would like to thank my fellow students and flat mates for making the stay in Stirling enjoyable time. Last but not least my supervisor for having effective meetings in which we were able to share knowledge a technical and economical level.

# Contents

[1. Introduction 7](#_Toc18356980)

[1.1 Background and Context 7](#_Toc18356981)

[1.2 Scope and Objectives 8](#_Toc18356982)

[1.3 Achievements 9](#_Toc18356983)

[1.4 Overview of Dissertation 10](#_Toc18356984)

[2. State-of-The-Art 11](#_Toc18356985)

[2.1 Workings of Blockchain 11](#_Toc18356986)

[2.2 Traditional insurance problems 14](#_Toc18356987)

[2.3 Blockchain based insurances 16](#_Toc18356988)

[2.4 State-of-The-Art applications 18](#_Toc18356989)

[Etherisc 18](#_Toc18356990)

[Nexus Mutual 19](#_Toc18356991)

[Teambrella 20](#_Toc18356992)

[Augur 20](#_Toc18356993)

[2.5 Potential cases 21](#_Toc18356994)

[Life insurance 21](#_Toc18356995)

[Flight delay insurance 21](#_Toc18356996)

[Earthquake Insurance 21](#_Toc18356997)

[Theft insurance on Bicycles and cars 22](#_Toc18356998)

[Focused use case (earthquake insurance) 22](#_Toc18356999)

[3. Earthquake insurance Dapp 23](#_Toc18357000)

[3.1 Infrastructure and third parties 23](#_Toc18357001)

[Blockchain platform 23](#_Toc18357002)

[Etherisc and GIF (Generic insurance framework) 23](#_Toc18357003)

[Provable 24](#_Toc18357004)

[3.2 Application requirements 26](#_Toc18357005)

[General application process 26](#_Toc18357006)

[User requirements 26](#_Toc18357007)

[Application process 27](#_Toc18357008)

[Claims process 28](#_Toc18357009)

[4. Conclusion 29](#_Toc18357010)

[4.1 Summary 29](#_Toc18357011)

[4.2 Evaluation 29](#_Toc18357012)

[4.3 Future work 30](#_Toc18357013)

[References 31](#_Toc18357014)

[Appendices 33](#_Toc18357015)

[Appendix 1: Project proposal 33](#_Toc18357016)

[Appendix 2: Example of Seismic portal data 34](#_Toc18357017)

[Appendix 3: List of user functions 34](#_Toc18357018)

[Appendix 4: Dockerfile Risk Assessment 35](#_Toc18357019)

[Appendix 5: Dockerfile ClaimVerification 35](#_Toc18357020)

[Appendix 6: risk.py 36](#_Toc18357021)

[Appendix 7: Claim.py 37](#_Toc18357022)

[Appendix 8: RiskAssessment.sol 38](#_Toc18357023)

[Appendix 9: ClaimVerification.sol 38](#_Toc18357024)

[Appendix 10: EarthquakeInsurance.sol 39](#_Toc18357025)

[Appendix 11: ImplementationEtherisc.sol 41](#_Toc18357026)

# Introduction

## 1.1 Background and Context

***How could a blockchain insurance application (Dapp) solve problems in traditional insurance?***

Insurance has been around for a long time. Although it has changed over the centuries. When insurance was first applied in smaller communities’ decisions would be made by voting or appointing a leader to decide. Through modern economy and globalisation have these smaller local communities become more connected with the rest of the world. This changed insurance as it has become more centralized in order to diversify risk. Diversification of risk is one of insurances main functions as it tries to spread and share risk over many insureds with different properties. These properties could for example be geographical locations, product brands or ages.

“Once people are insured, they become members of a solidarity group that goes far beyond their own circles. In the past, a family or village, maybe an extended family or a network of little villages could work together to share certain risks and compensate those that suffered from ill fortune.” [4] As Liedtke points out, in the past smaller communities worked together to share risk out of solidarity for their communities. This solidarity is now extended toward a global community. Although, this principle of solidarity has almost been forgotten in the name of creating larger and more risk-diverse communities. As result, people don’t always see or feel the benefits of the insurance community anymore. What if new technologies could help us to achieve a combination of both global sharing risk and solidarity within the insurance community?

## 1.2 Scope and Objectives

The objective of this dissertation is to create a prototype of an insurance product based on a blockchain. The prototype serves as a proof of concept for the validity of insurance products on a blockchain. Finding a suitable use-case will be part of the research. The focus will be on creating a working back-end with smart contracts for an insurance product. Third parties should be able to interact with the smart contracts and build a user-interface on top. For users to interact with the smart contract itself it requires a high level of technical skill compared with the average user. Therefore, this will be a limitation of the scope as the dissertation is not focused on the front-end and creation of a user-friendly interface.

The scope is limited on the economics behind insurance, as the long-tail-risk and capital requirements, fair risk assessment and pay-out conditions can be dissertation topics on themselves. Although, the final prototype will contain a basic approach to assess risk limited due to lack of data and risk expertise. The limitation due to lack of risk expertise and data should also be considered for the pay-out conditions.

The long-tail risk and capital requirements are issues on which Etherisc is trying to provide further expertise as the application will be built on top of Etherisc in the future these matters will be out of scope. In addition, regulation will be out of scope due the fact that insurance regulation varies a lot between countries and can be wildly complex. Still many discussions are held over the validity of smart contracts within legal systems as regulation is slow to adopt new technologies. Regulation should still be a major concern as the insurance industry is highly regulated and different areas of regulation. For example, GDPR might influence choices within the design of the end-product.

## 1.3 Achievements

Succeeded to create working proof of concept of an earthquake insurance smart contract. Third parties are able to build an application on top of the smart contract which functions as the back-end. The smart contract is able to run two computation queries that are created with Dockerfile which run a Python script. Both scripts provide the query with a score. The queries require longitude and latitude coordinates that will be used in the application and claim phase. In the application phase the query will return a risk score calculated based on historical data of significant earthquakes. The queries are not fully integrated into the Etherisc GIF (generalized insurance framework). Although, a similar basic process flow has been used to make sure implantation in the future should succeed. In the process of application, a RiskScore is received by a query which will partly determine the premium of the applicant. The premium is returned to the user as a premium quote. The applicant applies for the policy. The underwriter will accept or reject the application. The longitude and latitude coordinates are then used in the claim query which verifies whether the claim leads to a (partially) pay-out. The application meets the user requirements set in chapter 3.2 Application requirements.

## 1.4 Overview of Dissertation

The background research will be conducted on how traditional insurance functions and the workings of a blockchain. With the help of previous researched background information, the generic problems in the traditional insurance industry will be to analysed. After having identified these issues, the potential of applying blockchain technology to the insurance problems will be further be examined. This will also include a general analysis of already existing applications on a blockchain trying to solve traditional insurance problems. To gain more focus several use-cases will be briefly described and their potential for a PoC (proof of concept) will be addressed. A use-case will be used to propose a smart contract solution for the back-end of a Dapp (Decentralized App). In this proposal the major focus will be on the back-end and therefore the functionality of the smart contracts. Application requirements will provide an overview of the application and user interaction. Finally, this should lead towards a functioning PoC. The description of the PoC will include the technical details on how the chosen use-case for insurance is translated into the deployed smart contracts.

# State-of-The-Art

## Workings of Blockchain

The concept of a blockchain was first introduced by an unknown writer called Satoshi Nakamoto. In this paper not only, the concept was introduced it also described one of its biggest and first applications namely Bitcoin. The title of the paper: A Peer-to-Peer Electronic Cash System. [5] The emphasis of this paper lies on the peer-to-peer system. Cash had always been a peer-to-peer method of transacting but transactions had always been physically. Through the convenience of digital money, the complexity of payment systems grew with the increasing number of intermediaries. The potential usage of blockchain has created the possibility to disintermediate current payment systems back toward decentralized peer-to-peer systems. Could the same technology be used to disintermediate other systems as insurance?

Blockchain technologies could best be described with the term earlier known distributed ledger technology. A distributed ledger is a shared record where all peers have an exact copy of the ledger. Any new records that enter the ledger via one of the participants should be approved through a protocol to achieve consensus, hence the name consensus protocol. The most used platforms use proof-of-work consensus protocol. Proof-of-work uses miners that will “mine transactions” in order to get a reward. Mined transactions are transactions put through the consensus protocol and recorded into the blockchain. The blockchain itself is the distributed ledger. Records are bundled into blocks, every block in the blockchain is hashed. The hash is added into the next block. Therefore, blocks will be recorded in a chain where it is impossible to change a previous block. Changing a previous transaction in a block would require consensus on all blocks recorded in the chain that are mined and following the protocol to start a new chain that overwrites past blocks. As long as the network remains decentralized the network is a robust peer-to-peer system in which transparency and disintermediation plays a major role.

Another additional impressive application that has been realised with including a peer-to-peer currency has been Ethereum. Bitcoin has restrictions due to its limited scripting language. It was designed to provide a safe and robust payment network. Ethereum on the other hand has language which is more flexible and therefore, could be considered less robust. As the Ethereum blockchain provides the possibility of running compiled Solidity code on a decentralized computer. This virtual computer is called the Ethereum virtual machine (EVM). Nick Szabo defined the term “smart contracts” for the first time in 1994 as “a smart contract as a computerized transaction protocol that executes terms of a contract.” [6] The term smart contract could be applied to transactions on the bitcoin network but also to the compiled solidity code stored in the Ethereum blockchain. Both platforms are capable of combining a store of value and information into computerized protocol. To review which platform could best be used for smart contracts on insurance the following platforms are further researched: Bitcoin, Ethereum, Rootstock. All of these platforms are public and have the potential to write smart contracts. The platforms have similarities but also differences in their properties. The similarities are mainly due to their evolutionary relationships.

Bitcoin is the biggest and most robust platform. One of the reasons for this is due to its limited scripting language which doesn’t allow much room for errors. Bitcoin is not Turing complete and therefore not able to handle much complexity within its smart contracts. In contradiction to Ethereum has as Vitalik Buterin mentions in the white paper the intention for Ethereum was to be Turing complete. “What Ethereum intends to provide is a blockchain with a built-in fully fledged Turing-complete programming language that can be used to create "contracts" that can be used to encode arbitrary state transition functions, allowing users to create any of the systems described above, as well as many others that we have not yet imagined, simply by writing up the logic in a few lines of code.” [7] Important to note is that Ethereum was created out of the frustration that bitcoin had Turing-incompleteness. Therefore, the Turing-completeness of Ethereum’s programming language plays a major role as it is a property which bitcoin intentionally refused to implement. When trying to decipher the closest readable language to machine language opcodes (operation codes) give us the ability to see what is going on a lower level language. The Bitcoin scripting language is closely related to opcodes. In contradiction to Ethereum, Bitcoin doesn’t consist of opcodes which have JUMP, JUMPI or JUMPEST. As is stated in the Ethereum white paper “The alternative to Turing-completeness is Turing-incompleteness, where JUMP and JUMPI do not exist and only one copy of each contract is allowed to exist in the call stack at any given time.” [7] Therefore, Bitcoin scripting language restricts the code to JUMP forward or backwards which makes it impossible to loop over the same code again. In order to prevent the code from running eternally through an eternal loop, Ethereum introduced Gas. Gas is a fee that each contract pays related to its computation it uses. Before a computation a certain amount of Gas is set, if it burns through all the Gas the computation will stop. Call functions don’t use any Gas as it doesn’t actually change the state of the contract it just reading a value from memory. By running code in the smart contract, the state of the contract changes and therefore Gas is being used to perform the computation on the EVM. The EVM is where the opcodes from the smart contracts are executed. Solidity, Serpent or Vyper are the higher-level languages in which smart contracts can be written. Solidity is the most commonly used. Solidity is based on JavaScript and is compiled to machine language in order to execute on the EVM.

Besides the main blockchains, second layer solutions are proposed. Second layer solution are solutions not imbedded in the blockchain itself, but on the level of an application. The second layer solutions are usually used as a communication layer. Not all blockchain provide the necessary infrastructure to suffice the need of the communication layer within the blockchain. Rootstock (RSK) is a second layer implementation trying to combine the stability of bitcoin and the flexibility the Ethereum scripting language. RSK is a sidechain of bitcoin therefore has the ability to use bitcoin in transactions. The smart contracts are executed on a VM (virtual machine) that is compatible with the EVM. Therefore, the same smart contract code can be used to launch on Ethereum and RSK. [8]

Most blockchains face a common problem of storing, accessing and computing large numbers of data. For example, it is impossible to create the truly same random number on all the same VM’s. It would also be very inefficient to store a large amount of random numbers onto all VM’s. Therefore, oracles are created which are data feed providers from outside the blockchain. These oracles provide queries to import data into a blockchain and use this in smart contracts. In the case of the previous example the data imported into the blockchain would be a random number calculated outside the blockchain. Oracles are a major help in order to breach the gap between the blockchain and physical real-world information. Although some argue that oracles should not be trusted due their centralization. For now, it seems to be the only solution to import information from outside the blockchain to trust a third party. Astraea is currently working to create a decentralized oracle to solve the centralized point-of-failure of currently used oracle solutions. [9] Currently one of the biggest providers of an oracle service is Provable (changed its name from Oraclize). [2] Provable provides a library contract which can be used to query from several sources.

Applications on a blockchain also called Dapps (decentralized apps) connect through a webservice interface. The developing of a Dapp is similar to that of an app or website. Although the Ethereum network will function as the back-end of the app which gives its decentralized properties. To explain how users, interact with the network figure 1 shows a diagram of the whole system. Users are able to deploy contracts within the network on to the Ethereum blockchain and can be accessed via one of the EVMs as they serve as access points. Front-end programs are able to connect to the Ethereum network through one of the EVMs by using one of the web3 libraries. When connected to the to the network users are able to interact with their wallets to sign contracts and transact.

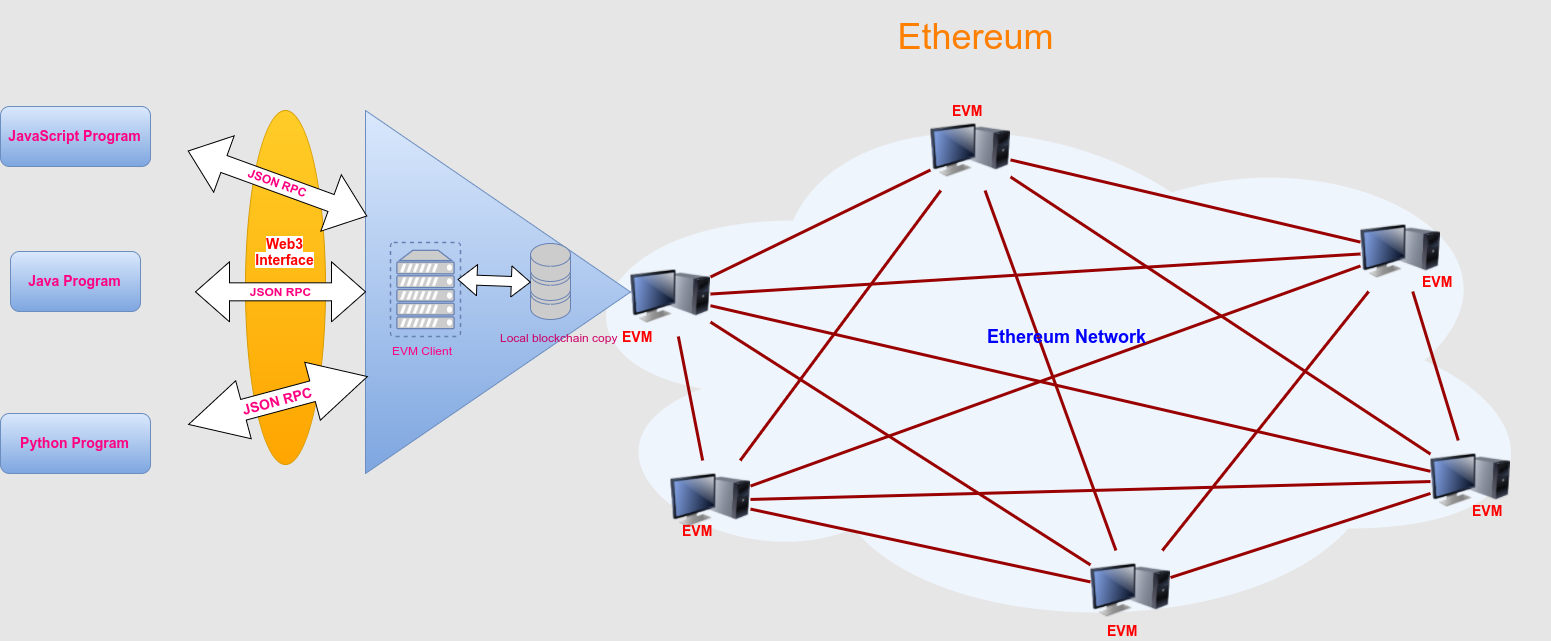


Figure Brief explanation of the Ethereum network and applications [10]

## Traditional insurance problems

In this chapter the current problems in the traditional insurance model are highlighted and how blockchain could help solve these problems.

The traditional insurance model could be considered as slow, mostly due to its process in which a lot of human activity is involved. The process might vary over the different insurance cases. In figure 2 you can see a generic version of information and money flows between a customer and an insurance company. From the customers perspective it looks two phased. First the phase of acquiring the insurance policy and premium and accepting the contract. The second phase could be considered as the phase where a customer claims it’s right to use the insurance according due to the policy requirements. Claims on an insured risk are made, assessed and claims are either payed-out of the insurance benefits or denied. The insurance is an intermediary that facilitates financial security against its insured risks.

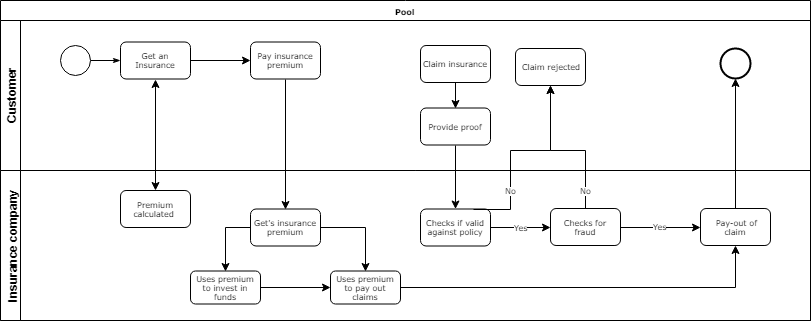


Figure Information and money flow inside a traditional insurance

As is pointed out in the book the future of fintech information asymmetry on policies plays a major role when it comes to sales of insurance. As the book mentions: “Currently, insurance company’s mainly focus on pushing for new deals and contracts, with customers who are passively recruited and are not fully aware of what they are signing up for.” [11] This begs the question of this is a sustainable business. This information asymmetry indicates that the insurer and the insurance agency don’t have the same information regarding the product. Therefore, the insurance company should strive to inform customers in an easily matter. Due to the complexity of product this is quite the challenge. Customers usually like being well informed if the information is convenient to obtain. This is especially the case for a boring product like an insurance policy.

This information symmetry often leads to a principle-agency problem where what is best for the customer is not aligned with the company’s interest. Where a broker’s incentives to take risk are not always aligned to the risk of what the customer could ultimately bare, the insurance company might oversell to a customer and try to deny claims. The relationship between an insured and the company should be full of trust. Instead this is not always the case, mainly due to the fact that insurance companies have a very risk-averse culture and are trying to protect the interests of the other participants. Which results in discussions about risk premiums and denied claims.

A Quora post highlights similar issues and two additional ones related that might be related. [12] Strangely as an intermediary most traditional insurers seem to have minimal customer interaction. Mainly due to the hierarchical distribution in which intermediaries are used to reach customers. The use of these long distribution models leads to high overhead cost and a greater distance to discover new customer needs. “There is a fear from technology in this industry.” [12] Technology is considered to be a risk in short term. Insurance companies don’t seem to like taking risk as its their core business is to assess and minimalize risk. Due to this risk aversion technology adoption lags behind. This might be explained by that culture including characteristic of its workforce.

Joshua Davis [13] argues that insurance is a heavily regulated industry and therefore it’s hard to enter the market. Interestingly, peer-to-peer lending communities do not have the same levels of regulation. It raises the question whether a peer-to-peer insurance by smart contracts is fundamentally different than a traditional insurance.

To conclude the major problems in the traditional insurance this short summary is provided in bullet points

* Information asymmetry hence principal-agency problem
* Transparency related to trust
* Minimal customer interaction therefore no insight to customer needs
* Lack adoption of technology due risk-averse culture and workforce
* Highly regulated (blessing and a curse)

## Blockchain based insurances

Blockchain is a distributed ledger which is able to store a transaction into its block. Peers have the exact same copy and communicate with a protocol to create new records without collision. When blocks are established due to the protocol of proof-of-work they can’t be changed. Every block is used to build a chain. The chain keeps growing and uses a hash (summary) of its previous block. Therefore, if you try to change one of the previous blocks the hash will change and the rest of the peers will refuse change because the majority will have a different (the real) copy of the ledger. Within a transaction it is possible to add code. This code is immutably saved within the blockchain and can be called by users. The code has the power to combine the flows of value and information within itself. This code would be considered as a smart contract. Blockchain and smart contracts have properties to create disintermediation and the possibility in achieving a more transparent and cost-efficient insurance. “The application of blockchain to the automation of contracts allows a reduction in administration costs for reconciliation and error. Smart contracts powered by a blockchain could provide customers and insurance companies with ways to manage claims in a transparent, responsive, and irrefutable manner.” [11] It’s still up to developers of smart contracts to translate the paper contracts into smart contracts in a safely, efficient and transparent matter. Additionally, Volosovich Svetlana describes a list where blockchain could solve current issues in traditional insurance.

“In the insurance sector blockchain technology will help to solve the following process automation tasks:

- creation and monitor of insurance history,

- operative access to certificates of the state bodies,

- insurance events accounting, their instant fixation,

- insurance contracts conclusion online,

- expertise and preliminary risk evaluation,

- interaction between brokers and sales network,

- transparent regulation and response to customer requests.” [14]

Most solutions relate to the earlier mentioned problems about transparency and the distribution. Although these solutions are still quite abstract, currently two kind of insurance types are proposed as blockchain solutions. One of the solutions is based on the idea of a mutual fund which relate to peer-to-peer lending structures where a community lends funds towards a participant in need of funds due to certain risks based on a vote consensus. In this solution human activity is not removed but switched to the participants of the community. The protocols, incentivization and policy rules are used to create a fair policy. Therefore, it is still possible to create a wide variety a of different models. The second well be spoken solution is parametric insurances. “Parametric insurance is a form of insurance where the pay-outs are determined not through a claims adjuster surveying the damage, but based on objective measures, such as the magnitude of a weather event.” [15] Parametric insurances are easy to apply due objective measures which can be automated. It strips away the complexity of insurance policies by focussing on a specific risk event. This kind of insurance will not work for every use-case. One of the most impressive parametric insurance cases on a blockchain was introduced on flight delays. [16] InsurETH introduced an insurance which used public data of flights to establish if your flight was delayed by a certain time. If the flight was sufficiently delayed the insurance would automatically compensate without the customer claiming. “The key to parametric insurance is finding objective indicators that can serve as an effective proxy for the type of loss being covered.” [15] Without an objective indicator insurance becomes too complex to be automated. It’s important to note that in parametric insurance trust is centralized on the provider of the data. Parametric insurance will eliminate almost all friction and cost of having an intermediary. The indicators will serve as a proxy, therefore the payment will most likely not match the exact cost of the damage caused. The use of an external data feed is called an oracle. Parametric insurance almost always makes use of an oracle.

## 2.4 State-of-The-Art applications

Part of the research will be to look at the already existing applications. For a new product to develop it is important to know what similar applications have already been launched. Most application are trying to solve similar problems in the broader sense but have different approaches. By reviewing most popular applications, great insights and possibly cooperation can be achieved. It is also interesting to see which challenges current platforms face.

### Etherisc

Etherisc inspires to be more than just an insurance. It inspires to be a platform where multiple specialized parties can work together to launch an insurance product. Etherisc has provided an overview of the Etherisc model in figure 3 on their website. The platform is focused on collaboration of different parties through their protocol. This platform mainly supports products for parametric insurances through the use of oracles. By creating an ecosystem for the development and launching of new insurance products on the Ethereum blockchain Etherisc brings participants together. Otherwise the landscape of insurance solutions on Ethereum would be more scattered. Parties would be working on solving the same problem and never reach the scale which insurance needs. Etherisc created a set of solidity contracts which serve as libraries. These contracts combined create the GIF(Generalized insurance framework). The GIF is very useful for the implementation of any insurance product on the Ethereum network. It provides a very general infrastructure of any insurance business process. The GIF will further be discussed in 3.1 Infrastructure and Third parties.

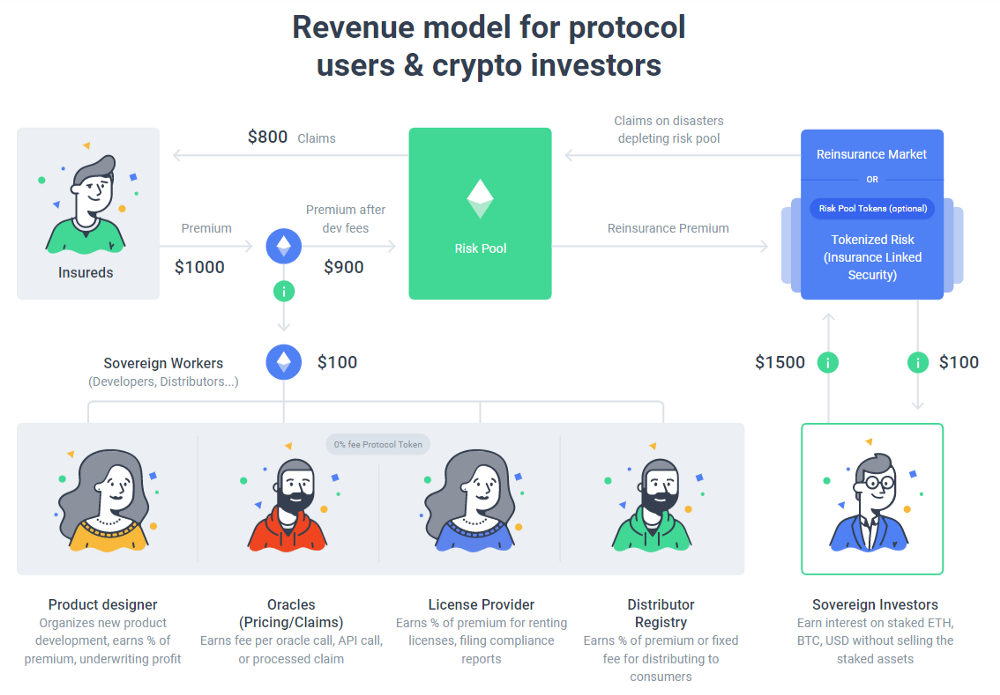


Figure Etherisc model [17]

Behind the scenes, Etherisc will use two type of tokens the DIP (protocol tokens) and RSC-FDD (Risk pool tokens). The DIP tokens are used by the insureds or sovereign workers as shown in figure 3. Insureds can use the tokens to buy insurance. The sovereign workers all have their own contributions to the protocol, for each role of the sovereign workers the tokens will be used in their own way. Product designers will be rewarded in tokens for the usage of the service. Oracles will be rewarded for providing reliable oracle data. Tokens can be staked to provide the required capital for a licence provider. The distributor registries are called sales agents which are rewarded for the distribution of the products by fixed number of tokens or sharing in the revenue and profits. Risk pool tokens are used to attract capital and thus fur fill the role of reinsuring. Sovereign investors also shown in will stake their investments into the RSC-FDD as they are essentially providing capital to insure someone else’s risks. The Sovereign investors will receive a compensation for providing the capital. “When a policy expires without a claim, its premium becomes revenue and it is allocated as follows:

1. 10% is reserved in the risk pool to subsidize premiums.

2. 20% is paid to the reinsurance pool to subsidize long-tail risk collateral

3. 70% is paid pro rata to the holders of RSC-FDD tokens as dividends.”

The distribution of the revenue will mainly flow towards the sovereign investors. Besides the revenue will not directly be returned towards the insured who didn’t claim. The revenue will be used to subsidize future premiums. Although this will only be 10% of the revenue whereas 70% will flow towards the sovereign investors. The last 20% will flow towards a reserve which is supposed to function as a collateral of the long-tail risk. Tail risk refers to extremely rare events where normal capital reserves are not sufficient enough to cover the impact of these rare events (for example massive earthquakes, hurricanes, of floods).

### Nexus Mutual

As earlier mentioned in Blockchain based insurances, two possible models are for insurance are used within smart contracts. Nexus Mutual’s name might indicate which of the two models they use. Nexus Mutual is rejecting the idea of parametric insurance due the possibility of unfair basis risk, oracle failure and limited products. Blockchain create the possibility to create peer-to-peer communities without the need of intermediaries. Insurance started this but through risk diversification and growth of these communities expanded to profit making corporates. As Nexus Mutual states in their white paper “Roughly 35% of insurance premiums are lost due to frictional costs in the system. Only 65% of premiums are returned to customers via claims, the rest is lost in distribution, operational expenses (including regulatory), capital costs and profit.” [18] The friction of the intermediary is costly and could be improved through automation in blockchain and smart contracts. Nexus Mutual estimates to reduce the friction cost of 35% by 18% through applying their model.

Members can buy ERC-20 tokens, these provide the funds for coverage. The price of the token will vary with the funding levels of the capital pool. The price will also be influenced by the amount required coverage and the existing capital in the pool funds. With this ERC-20 token Nexus Mutual is trying to create a supply and demand market for capital and thus incentivize members to supply capital when is needed. In this process they will take their minimum capital requirements as a base. The bought tokens can be used to purchase cover by burning the tokens. Similar members can contribute to claims and risk assessment in order to do these stakes are required to disincentivize fraud. By contributing to these assessments, the member is being awarded with tokens in return.

Nexus Mutual is launching its first product which will be a Smart contract cover. The cover will insure against unintended code usage. Etherisc and Nexus mutual started a collaboration this potentially promises new insurances products for both parties. [19]

### Teambrella

Similar to Nexus Mutual, Teambrella is a structure like a mutual fund where peers have the power to vote on decisions. The total premium amount is not fixed because peers pay directly to other peers when claims are approved. “The framework for the P2P organization includes a decision-making layer and a payment layer. The payment layer is based on blockchain technology. The decision-making layer consists of a server(s) used as a medium for communication and voting.” [20] Teambrella reveals in this statement that it is not fully based on a blockchain. A server is used to provide a decision-making layer which could be considered as a potential point of exploitation. Teambrella is supporting the two major platforms Bitcoin and Ethereum which will be the implemented payment layer.

Teams are can be made by anyone and have an initial set of rules. These teams’ function as communities where a particular interest related to the insurance forms social cohesion. Therefore, changes of peers to consider fraud are lowered. For every new peer a wallet is opened in which the peer has access via the private key. Consigners have access via a server which selects consigners that have to come to a consensus. Risk is assessment is applied when any new peer joins a team. Both the new peer and the existing peers of the team agree on a risk coefficient which set by rules or estimated on public data. This coefficient is a representation of to how much risk the new peer is exposed. The lower the coefficient the less the peer has to contribute towards claims. Peers are also allowed to set a coverage limit which is insuring only for a partially amount. Peers pay directly towards the peer that claimed. Which decentralizes the entire capital pool. Instead the capital resides in the wallet of the users. Therefore, liquidity is maintained for the peers. Although it makes managing the capital harder, the peer still needs to have sufficient funds in their wallet on the Teambrella application to participate.

### Augur

One might argue that insurance parametric insurance is like betting on events to happen or not. Augur is a prediction market where anyone can propose a bet on an event with a verifiable nature. This might be an alternative to traditional insurance models. Although in practise the application misses a large customer base behind a certain bet/insurance policy. Therefore, this might result in high cost due to the lack of risk diversification. It seems to be a more liberalistic approach which would provide more freedom to specify its requirements. But this could also lead to the overwhelming grow of customized policies based on information asymmetry. [21]

## 2.5 Potential cases

After analysing the some of the already existing applications it is time to look into a specific use case. One of the main challenges in finding the right use case is converting the logic of current insurances into smart contract code. The use cases are dependent on a generalised type of insurance as earlier discussed. In the white paper of Nexus Mutual they mention the two major models for claim assessment in insurance which are applied to smart contracts. [18] Firstly, parametric insurance which requires external oracles (data feed). Parametric insurance has a very strict set of rules based on its data from an oracle. This model is limits use cases due to the dependency on the external data feed. This creates issues revolving around trusting a centralized entity and its availability. Mutual fund insurance which is more community based. This community-based approach needs an active community in order to function. Every member of the community is able to vote on the two major processes. First, risk assessment where an applicant applies a policy to join the community. The community votes either to accept or reject the new applicant. The mutual fund structure is based how insurance functioned before insurance became centralized profit-orientated organisations, truly peer-to-peer. The complexity of a mutual fund structured insurance is much higher compared to parametric due to its human interaction and malicious behaviour. Therefore, the structure of a parametric insurance will be applied to a fitting use case. A few potential use cases are shortly described, one of these use cases will be the focus of the further development for a prototype.

### Life insurance

Almost every country has a register on which deaths of citizens are registered. For this use case death registries would be served as an oracle. Unfortunately, these registries are not public and therefore require a collaboration with the responsible organisation. When a family member dies, it is desirable that once life insurance claim is handled in a seamlessly matter. Family members are often still mourning and busy with organising the burial and dividing the inheritance which often leads to high emotional tensions. Imagine that the insurance funds are automatically allocated towards the remaining family members. This way a burden is taken away from the from the mourning family members. This solution could have a great impact on people’s lives. The use case still has a few barriers namely, access to the national death registries of the countries. This major barrier is driven by privacy laws that are not allowed to share this highly sensitive personal information.

### Flight delay insurance

Flight delay insurance has already been built on the blockchain. It turned out to be the perfect use-case for a blockchain implemented solution. The flight delays use a public register of flights as an oracle to verify if the insurer has a significant delay. Although, this is not a new use case and it has already been proven to be a good match for parametric insurance.

### Earthquake Insurance

Earthquakes can cause enormous damage to property in a concentrated area with low predictability of happening. Therefore, there is a need to share the risk globally. Earthquakes happen mostly at specific areas therefore it helps to diversify the risk over peers living in different high-risk areas. To identify if an earthquake has happened an oracle can be used. The Seismic Portal API could be used as an oracle which shows the latest earthquakes around the world including its magnitude, location and the date including time. [22] Even with all these indicators the challenge of parametric insurance is to estimate the financial impact of the damage. It is hard to identify the exact impact of the earthquake on the insurer’s property with currently given data. It is therefore more likely to create an event-based insurance where the insured decides its coverage of the property. Premium would be paid in ratio to the level of exposed risk and pay-out of claims accordingly towards indicators of the impact.

### Theft insurance on Bicycles and cars

The Netherlands holds more bicycles than people. Bicycles are used as a normal form of transport in the day-to-day life of most Dutch people. As your bicycle gets stolen it has a major impact on your mobility. In addition, it will take you time and money to buy a new bicycle. Which could be a very upsetting experience and always happen at a very unconvinced time. Most bicycles are currently uninsured, due to their low value and provability of it being stolen. Although, there is an upcoming trend of people buying e-bicycles which are electric driven bicycles. These bicycles are more often insured due to their high value. One of the most common problems is that it is hard to prove if the bicycle was locked during the theft, this is mostly proven through showing the its key. Otherwise the insured is being neglectful and therefore will not be eligible to claim. In the Netherlands RDW is providing information on bicycles that have been registered and stolen. [23] A pre-requirement for an insurance would be to register your bicycle to this register. If the bicycle gets stolen and you report it to the police it will appear as stolen in the register. This could be used to create a parametric insurance where coverage is determined by the insured and claims will be automated. Assuming that filing false police reports would be trackable and punishable. Similar to the bicycle use-case does RDW provide information about cars when searched on licence plate number. [24] This information contains whether the car is stolen or not. It also contains technical details which could indicate the value or the risk of the car.

### Focused use case (earthquake insurance)

The use cases earthquake insurance seems to have a great potential as a parametric insurance. Mostly due to its public and uptodate data. The oracle provides data about recently measured earthquakes. As is pointed out by Sarah Zielinski the impact of earthquakes relates to seven factors. [25] Some of these are more complex to capture by data, but a few are already available. Obviously, the location, magnitude, depth and distance to the epicentre (point right above the earthquake’s origin). An example of data provided by the Seismic Portal is shown in appendix 2. All of these are captured by some of the data provided by the oracle. Although other mentioned factors as geologic conditions (type of ground), aftermaths and architecture of buildings have more complexity and definitely a relation towards the impact of an earthquake. Therefore, these factors will be excluded in the product produced by this study.

Aside from the oracle data, users will need to provide data in order to assess the risk and verify claims automatically.

# Earthquake insurance Dapp

The earthquake insurance will ultimately be deployed on already existing infrastructure of Etherisc. As for the first prototype a basic infrastructure will be created to demonstrate the validity of the use case. The infrastructure does not include a front-end Dapp (Decentralized apps) although it provides a back-end foundation of a generalized insurance process. The result of this chapter will be a functioning back-end. Within the generalized structure by customizing the and implementing oracle queries the application will still be able to fully function with the required user input. Thus, the focus will be on the possibilities of smart contracts within the insurance industry. This research will focus on an earthquake insurance although different use cases where considered in the chapter 2.5 Potential cases.

## 3.1 Infrastructure and third parties

### Blockchain platform

As earlier discovered in this paper, Bitcoin is a platform focused on payments with a very limited flow of information and complexity. Therefore, Bitcoin’s platform doesn’t have the sufficient properties for creating a smart contract that would allow the complexity of an insurance policy. Ethereum on the other hand has Turing-completeness which allows for the creation loops and therefore more complexity. To stop code from running eternally Ethereum has introduced Gas as a fee for computation on the EVM. RSK might help to bridge the two platforms Bitcoin and Ethereum, by creating the possibility to execute the similar compiled smart contract code on a VM which uses a bitcoin sidechain. For this project it seems most fitting to choose Ethereum due to its possibility to scale and availability of documentation. Ethereum is currently the biggest platform which is Turing-complete and is the second biggest in market capitalisation. Besides, RSK could be used to expand the application to the users loyal to Bitcoin. Therefore, the biggest two cryptocurrencies could use the application. The smart contracts will be written in Solidity and tested on Ethereum. Rinkby test network will be used as spending Ether from the main network would be expensive and unwise due to the fact that others are able to interact with untested versions of the contracts.

### Etherisc and GIF (Generic insurance framework)

Etherisc presents an infrastructure called GIF. The platform consists of pre-written and standardized smart contract functions which facilitate the building of insurance products on Ethereum. The platform provides building blocks of a general insurance process shown in figure 4 .These building blocks are created for third parties to create their own products on top, instead of building the whole insurance process from the ground up. Although, this provides us with the major building blocks of our product the generic framework some customization will be made.

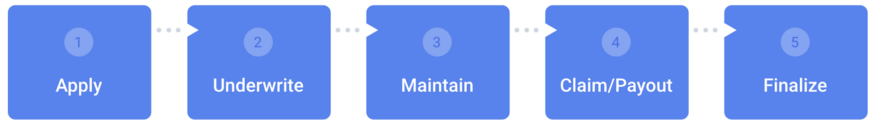


Figure General insurance process [26]

In figure 5 the same general process is shown with including the possible outcomes of a user interacting with any given policy. To give a short description a user can apply for an insurance policy, the underwriter can accept the application with a risk assessment. As long as the policy is active the user is able to submit. The claim can either automatically be verified or still be further assessed for the payed-out. The framework doesn’t have to be a parametric insurance. The structure could also be applied to a traditional or mutual fund insurance. It can still have manual verification of the claim by an independent expert or community. Although, this would slow down the claim verification process significantly. But Etherisc could be a solution for non-parametric insurance processes as well.

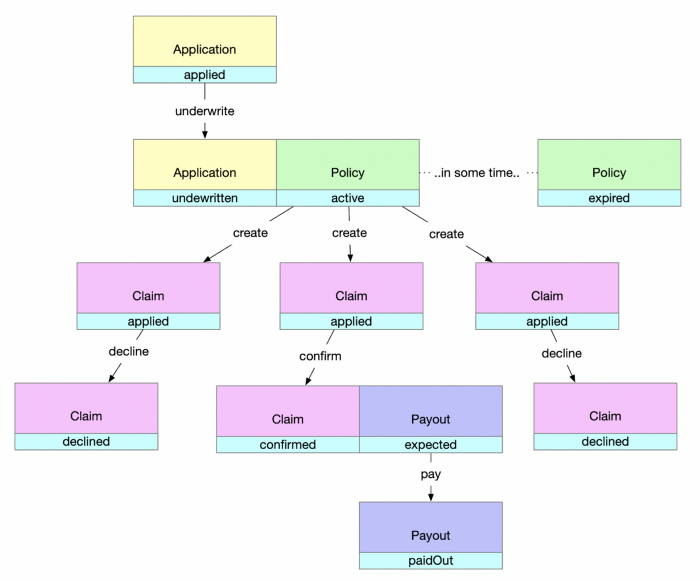


Figure Decision tree diagram of Etheriscs insurance process [26]

Etherisc has provided supporting tutorials on using their framework. [3] These seemed to be exceptionally useful in order to understand and use their infrastructure. The tutorial uses an open sandbox which can be used to develop and test products. After using both tutorials it still seems quite hard to fully implement within the new product on solidity.

### Provable

Provable provides an infrastructure to import data into the Ethereum blockchain. For lots of reasons having an external data feed is needed to make decisions. In this application the external data feed will be used to import data of new earthquakes and assess the risk of new applicants. Provable provides a few different methods to use queries. For example, an API URL can be used to retrieve a string or value. Although, in some situations there is a need for manipulation or calculation of the data. Provable also provides possibility to perform a computation query. The computation query is able to execute a script and import it’s results into the Ethereum blockchain. Provable documentation and code were used as a basis creating the queries. [2] The script files are shared via IPFS which is a decentralized storage system for files. The right environment was set up with the help of the documentation provided by IPFS. [27] When the file is uploaded to the IPFS network it receives its unique hash and therefore can be accessed by anyone and run. The hash guaranties that the files on the IPFS network cannot be tempered with as that would result into a different hash and therefore a different file. The current files are published with the following hashes on the IPFS network and freely accessible which provides transparency to the risk assessment and claim verification process.

* Risk Assessment = Qmb4kYSAHJgBFwMQekihpTn7cjiGxnZbzRvbN8t9RZ57k8
* ClaimVerification = QmQEpCpx23bLKG7YRJT5fE5ymr2nRvCnvj6KP9GocqUiRY

Provable is able to download and execute the files through the provided hash. The files contain a Docker file that is used to set the execution environment which runs a Python script to do the computations and data manipulations. The query supports user input from the smart contract and these variables are set as environment variables in order to be accessed by the scripts in the environment set by the Docker file. The code of the risk assessment Docker file and Python script are shown in appendix 4 and 6. The code of the claim verification Docker file and Python script are shown in appendix 5 and 7.

## 3.2 Application requirements

### General application process

As seen in the framework of Etherisc the general process of insurance exists in several phases of which the application and claim phase have interaction with the user. These two major phases will be the core of the application. In the application phase the user requests to join an insurance policy, before the application can be send the user will require a risk assessment. The risk assessment will determine the risk and therefore the premium of the applicant. The complexity of risk assessment is a challenge within smart contracts as the use of computation is costly. Computation queries will help to overcome this challenge. Within the application phase a computation query will be called based on the longitude and latitude of the applicant. The query will return a risk score to determine the premium in combination with the requested coverage. After receiving the premium quote, the applicant submits the application which contain the location and premium of the applicant. When the applicant is accepted by the underwriter. When the policy is active the insured is able to submit claims. The claims will be automatically be verified if the oracle returns a positive claim score. When the claim is verified the owner of the policy is verifies or rejects the pay-out as a manual check. The two different oracles will be used for risk assessment and claims. Both queries need different input data and therefore different queries.

### User requirements

The user requirements cover interactions with the user and thus set requirements for its functionality as a whole. The product that will be build is a generic earthquake insurance. The application should be able to obtain an insurance policy based on its location and self-determined coverage which verifies claims. This will be specified more in depth in the following user requirements for the application.

* To obtain a risk premium based on historic earthquakes.
* Submit an application with given risk premium.
* The underwriter can accept or reject applications.
* Manually submit a claim based on external data feed (oracle).
* Verify claims based on external data feed (oracle).
* The policy owner is able to verify or reject pay-outs.
* # Claims submitted by trigger oracle.
* # Payments of: Pay premium / pay-out claims

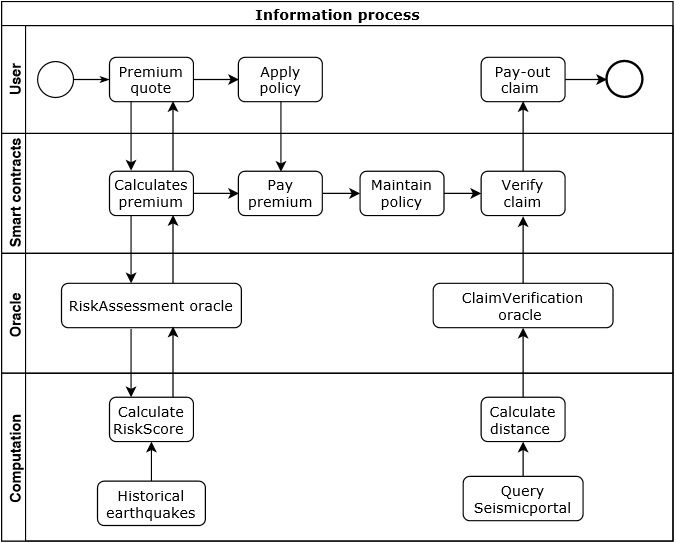


Figure Information flow diagram: The user requirements marked with ‘#’ are incorporated into the design, but likely not incorporated into the first prototype.

In figure 6 the process of the application based on ultimate user requirements is shown in a diagram. This diagram has three components: the user represents the front-end, the smart contract interacts with the provable. Provable is the oracle that creates a computation query in order to do off-chain computation for data manipulation. The computation files will be shared on IPFS towards Provable. Etherisc provides the general framework in which the two oracle queries can be integrated. Automating the claim query to create a claim without user input will be out of the scope of this project. The code for the general smart contract in solidity is shown in appendix 10.

### Application process

In the application process a user is able to request a risk premium based on its location. The risk assessment which will return the RiskScore which helps to estimate the risk premium (the value of risk). The risk premium will be calculated with a historical dataset with significant earthquakes from 1965 till 2016. This dataset was provided on Kaggle by the US Geological Survey. [28] The dataset provides a base on which longitude and latitude can be used to calculate the distance between the applicant and historical earthquakes. The haversine formula used to calculate the distance between an earthquake and an applicant. [1] The distance will be limited to 1600km as is assumed that all magnitudes have no impact out of this range. UP Seis defined different categories based off the magnitude indicator. [29] These categories are used to decide the impact of a past earthquakes. For each category of magnitude and distance every earthquake is assigned a RiskScore. After calculating the RiskScore for each earthquake met to the conditions of being within a distance of 1600km the average of RiskScores will be taken. The following scores will be assigned for the following conditions:

* Moderate 10 = Distance smaller than 32km and Magnitude between 5.0 - 6.0
* Strong 30 = Distance smaller than 80km and Magnitude between 6.0 - 7.0
* Major 80 = Distance smaller than 160km and Magnitude between 7.0 - 8.0
* Great 100 = Distance smaller than 1600km and Magnitude between 8.0 - 10.0

If no earthquake is within 1600km the RiskScore will be set to 1. When a RiskScore is obtained, the RiskScore will be used to calculate the premium. The premium will be the coverage multiplied by the RiskScore divided by 1000 to get a realistic premium for the risk of the insured. This risk assessment model will most likely not suffice to make proper risk assessment. As it is still untested, build on top of a lot of assumptions and quite basic. The model might be expanded with more features provided by the user as soil and construction type. To expand even further a classification machine learning algorithm could improve the risk assessment and claims verification even further. Although, the sufficient expertise of geology could vastly improve the set of the current rules. The code of the risk assessment query in solidity is shown in appendix 8.

### Claims process

The claim process starts whenever an insured policy meets the pay-out conditions of the policy. This is initiated by the insured, by submitting a claim. By letting the user submit a claim the frequency and therefore gas cost would be lower as the query has quite a high gas cost. The user will be able to manually submit a claim that will be verified by the query which returns the ClaimScore. The ClaimScore has the same conditions as the categories used for the RiskScores. The Seismic Portal provides an API to access a public data feed of measured earthquakes on a global scope. [30] The ClaimScore also uses the haversine formula to calculate the distance between the current earthquakes of a magnitude of 5.0 or higher. If the distance and magnitude meet any of the category conditions a ClaimScore is provided. If the earthquake doesn’t meet any of the conditions the ClaimScore will be set to 0. The ClaimScore is used to in order to determine the amount of pay-out towards the insured. For now, claims with a score above 0 will be verified. Although further development pay-outs of claims will be determined by multiplying the ClaimScore by the set coverage. For example, if the ClaimScore is 10 the owner of the insurance will receive 10% of the coverage amount. This is not necessarily tested, but shows the capabilities of smart contracts interacting with computation scripts. The code of the claim verification query in solidity is shown in appendix 9.

# Conclusion

## 4.1 Summary

Insurance has become larger than ever, mainly due to the effects of risk diversification. Although, it seems that having a large centralized intermediary might not be the most efficient way to apply insurance. Blockchain technology has decentralization properties which can be used to create insurance products in a decentralized matter and disintermediate the current industry. Although, not every blockchain has the capability of performing such complex tasks. Ethereum does have these capabilities and is therefore used. A few use cases where considered to create a proof of concept. Earthquake insurance seemed the most applicable use-case due to availability of data and the potential to implement as a parametric insurance. Etherisc provides a smart contract which can function as basis to build new insurance products on top. This framework is called GIF(Generalized Insurance Framework). Besides Etherisc, Provable provides a similar framework to use oracle queries. Provable will be used to import data into the Ethereum blockchain. The application has a few general phases which are derived from Etherisc’s GIF. The earthquake insurance starts with an application and premium quote. To receive a premium quote a risk assessment will be made through a computation query. The query accesses a python script via IPFS which is run on a virtual machine. The result will be a RiskScore and is then imported back to the blockchain. The score is used to calculate the premium. When the premium is received an application can be submitted. The underwriter is able to decline or accept the application. When accepted the policy will be active and claims can be submitted. A claim can be submitted and verified by an oracle that checks the location and magnitude of the latest earthquake resulting in a positive or negative ClaimScore. If positive the claim will be verified and the pay-out amount is equal to the coverage. Before the pay-out a manual check will decide whether to accepted or rejected.

## 4.2 Evaluation

The application does function on a basic level. As the implementation of the Etherisc framework GIF was not yet finished. The application is able to have functions on two levels owner of the contract and any interacting customer. The customer is able to request a price premium a based on its RiskScore and apply for an active policy. Payment of the premium and payment of pay-out is still restricted as this premium should be adjusted to a reasonable level to test these functions with test Ether. Normally, this would be manged by Etherisc’s framework. As implementation will be a scheduled for future work effort of recreating such infrastructure was reserved. Besides, this the basic application is able to create a process flow restricted by ID input which interacts with the two oracle queries in order to assess risk and verify claims initiated by manual input. The queries still can take up to 5 mins to calculate risk and claim scores. For users 5 min will be too long to wait for receiving a quote via the application. Therefore, more effective ways of risk assessment could be proposed. Security of smart contracts is usually a major concern, on the development of this code security was not central for the design. Therefore, a validation of the security of the contract is necessary.

## Future work

As Etherisc’s platform GIF is mentioned the current application is still not implemented in the GIF. This will be the first and major concern for any further development. GIF provides a safe main infrastructure for the application and in therefore important for further development of the application. It takes care of many aspects of insurance which might complicate the development of products separately. To improve customer value claim queries should be triggered by returning thresholds on a timely basis. If thresholds are reached policies will create their own claims based on the set requirements. Solidity code for resuming work on implementation is shown in appendix 11.

The data used might be expanded with more features provided by the user as soil and construction type. These features could help the risk assessment and claim verification as they are major contributing factors. Instead of the current risk assessment method a classification machine learning algorithm could be applied to improve the speed and performance of the risk assessment. Machine learning could also be applied to verify claims and their amount. Although, this would require a lot of training data label with accurate amounts of pay-outs. The current risk assessment and claim verification could already be vastly improved by sufficient expertise of geology. The thresholds of magnitude, distance and depth should be adjusted by experts as this requires a lot of expertise of geology. To refine the accuracy of the risk model and claim model classification of soil and classification of the building construction could be added as these factors would influence the impact of an earthquake. The conditions for claims should be set and adjusted accordingly. For now, these thresholds will be set to a default according to assumptions.

To create an interactive user interface for the smart contract functions. Common users have quite a low understanding of using blockchain smart contracts applications without any sensible user interface. Besides, it’s possible to implement logic into the front-end side of an application which is too costly to put into Solidity. As JavaScript does not have any limitations on usage of gas. When developing the front-end security, legal and privacy issues should be taken into consideration.

# References

|  |  |
| --- | --- |
| [1] | M. Dunn, “Haversine Formula in Python (Bearing and Distance between two GPS points),” 09 02 2015. [Online]. Available: https://stackoverflow.com/questions/4913349/haversine-formula-in-python-bearing-and-distance-between-two-gps-points#4913653. |
| [2] | Provable, “Documentation,” Provable, 28 08 2019. [Online]. Available: https://docs.provable.xyz/. [Accessed 01 09 2019]. |
| [3] | Etherisc, “GIF Tutorial: Part 1 — How to Build a Simple Insurance Product,” Etherisc, 14 04 2019. [Online]. Available: https://blog.etherisc.com/gif-tutorial-part-1-dc595057c1b9. |
| [4] | P. M. Liedtke, “What’s Insurance to a Modern Economy?,” *The Geneva Papers,* vol. 32, pp. 211-221, 2007. |
| [5] | S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” 2008. |
| [6] | N. Szabo, “Smart Contracts,” 1994. |
| [7] | V. Buterin, “Ethereum White Paper”. |
| [8] | S. D. Lerner, “RSK White Paper Overview,” 2019. |
| [9] | J. Adler, R. Berryhill, A. Veneris, Z. Poulos, N. Veira and A. Kastania, “ASTRAEA: A Decentralized Blockchain Oracl,” in *Cybermatics*, Athens, 2018. |
| [10] | P. Ramanujam, “iotbl,” iotbl, 10 03 2017. [Online]. Available: https://iotbl.blogspot.com/2017/03/ethereum-and-blockchain-2.html. |
| [11] | B. Nicoletti, “A Business Model for Insurtech Initiatives,” in *The Future of FinTech*, 2017, pp. 211-249. |
| [12] | N. Gupta, “What are some major challenges in the insurance industry?,” 09 01 2017. [Online]. Available: https://www.quora.com/What-are-some-major-challenges-in-the-insurance-industry. [Accessed 27 06 2019]. |
| [13] | J. Davis, “PEER TO PEER INSURANCE ON AN ETHEREUM BLOCKCHAIN”. |
| [14] | V. Svetlana , “INSURTECH: CHALLEGES AND DEVELOPMENT PERSPECTIVES,” *International Journal of Innovative Technologies in Economy ,* pp. 39-42, 2016. |
| [15] | A. Cohn, T. West and C. Parker, “SMART AFTER ALL: BLOCKCHAIN, SMART CONTRACTS, PARAMETRIC INSURANCE, AND SMART ENERGY GRIDS,” *GEORGETOWN LAW TECHNOLOGY REVIEW,* pp. 273-303, 2017. |
| [16] | InsurETH, “SMART FLIGHT INSURANCE,” Londen, 2015. |
| [17] | Etherisc, “Etherisc,” Etherisc, [Online]. Available: https://etherisc.com/. [Accessed 05 07 2019]. |
| [18] | H. Karp and M. Reinis, “NEXUS MUTUAL”. |
| [19] | Etherisc, “Etherisc and Nexus Mutual — bonus tokens for DIP holders,” Etherisc, 14 06 2019. [Online]. Available: https://blog.etherisc.com/etherisc-and-nexus-mutual-bonus-tokens-for-dip-holders-a6901cfd69f3. [Accessed 02 07 2019]. |
| [20] | A. Paperno, V. Kravchuk and E. Porubaev, “Teambrella: A Peer-to-Peer Coverage System”. |
| [21] | J. Peterson, J. Krug, M. Zoltu, A. K. Williams and S. Alexander, “Augur: a Decentralized Oracle and Prediction Market Platform,” 2018. |
| [22] | Seismicportal, “Seismicportal,” [Online]. Available: https://www.seismicportal.eu/. [Accessed 01 09 2019]. |
| [23] | RDW, “Fietsdiefstalregister,” RDW, 2014. [Online]. Available: https://fdr.rdw.nl/. [Accessed 06 07 2019]. |
| [24] | RDW, “Kentekencheck,” RDW, [Online]. Available: https://ovi.rdw.nl/default.aspx. [Accessed 2019 07 06]. |
| [25] | S. Zielinski, “smithsonianmag,” 23 02 2011. [Online]. Available: https://www.smithsonianmag.com/science-nature/seven-factors-that-contribute-to-the-destructiveness-of-an-earthquake-44395116/. [Accessed 11 07 2019]. |
| [26] | A. Konstantin, “GIF Tutorial (Part 2): The Policy Module,” Etherisc, 04 07 2019. [Online]. Available: https://blog.etherisc.com/gif-tutorial-part-2-the-policy-module-3da5ef3fb7fd. |
| [27] | IPFS, “IPFS Documentation,” IPFS, [Online]. Available: https://docs.ipfs.io/. |
| [28] | U. G. Survey, “Significant Earthquakes, 1965-2016,” 2016. [Online]. Available: https://www.kaggle.com/usgs/earthquake-database/kernels. |
| [29] | “Earthquake Magnitude Scale,” [Online]. Available: http://www.geo.mtu.edu/UPSeis/magnitude.html. |
| [30] | S. Portal, “Seismic Portal API,” Seismic Portal, [Online]. Available: http://www.seismicportal.eu/fdsnws/event/1/query?limit=1&format=json&minmag=5.0. |

# Appendices

## Appendix 1: Project proposal

*Overview:*

The goal is to examine and build a decentralized insurance based on a blockchain. The insurance policy will be based on smart contracts. One of the main challenges will be converting the business logic into a smart contract. This could be explored for different insurance cases. These insurances might be deployed on the already existing Dapps(Decentralized apps) or on a simple created one. Part of the research will be to look at the already existing applications and create one on an existing platform. The focus will be on the possibilities of smart contracts in the insurance industry. This will also include the architecture of the financial incentives of users.

*Timeline:*

1. Find a potential insurance case for insurance (Weeks 1-3)
2. Explore already existing applications (Weeks 1-3)
3. Learn the smart contract languages (solidity and Vyper) (Weeks 1-5)
4. Create the architecture of the application (Weeks 3-5)
5. Create Smart contracts (Weeks 5 – 8)
6. Deploy and test smart contracts (Weeks 9 -10)
7. Finalize the documentation and report (Weeks 6 -12)

**List your hardware and software requirements**

Hardware:

* Laptop or PC

(Potentially) software:

* Remix (Solidity IDE) available in browser. <https://remix.ethereum.org>
* Vyper (Vyper IDE) available in browser. <https://vyper.online/>
* Ganache (local test network Ethereum).
* Metamask wallet (chrome extension).

## Appendix 2: Example of Seismic portal data



## Appendix 3: List of user functions

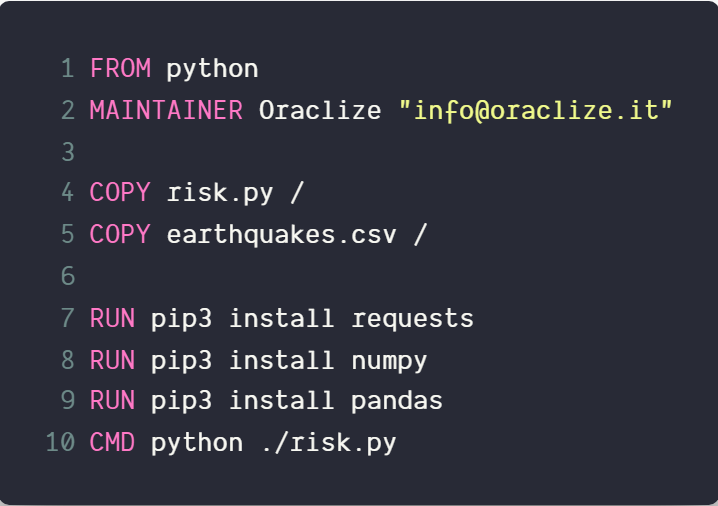
**Application phase**

* setRiskScore (Any)
* getQuote (Any)
* applyForPolicy (Any)
* underwriteApplication OR declineApplication (Only Owner)

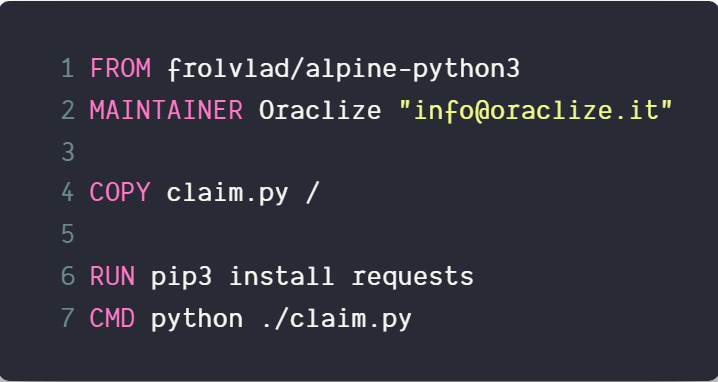
**Claim phase**

* setClaimScore (Any)
* createClaim (Any)
* confirmClaim (Only Owner)
* confirmPayout (Only Owner)

## Appendix 4: Dockerfile Risk Assessment



## Appendix 5: Dockerfile ClaimVerification



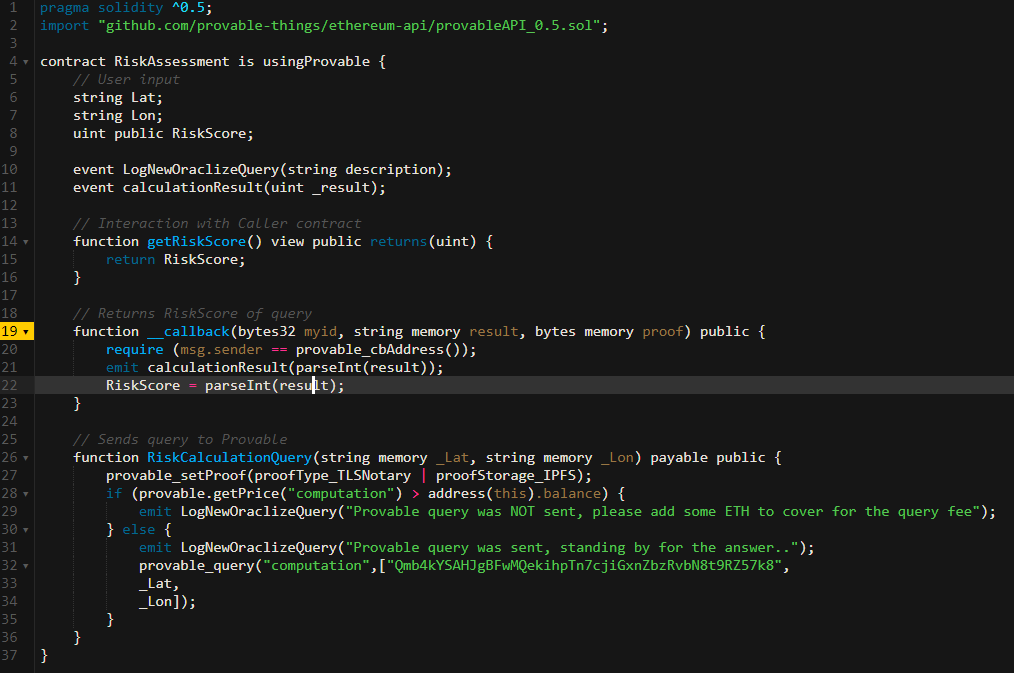
## Appendix 6: risk.py



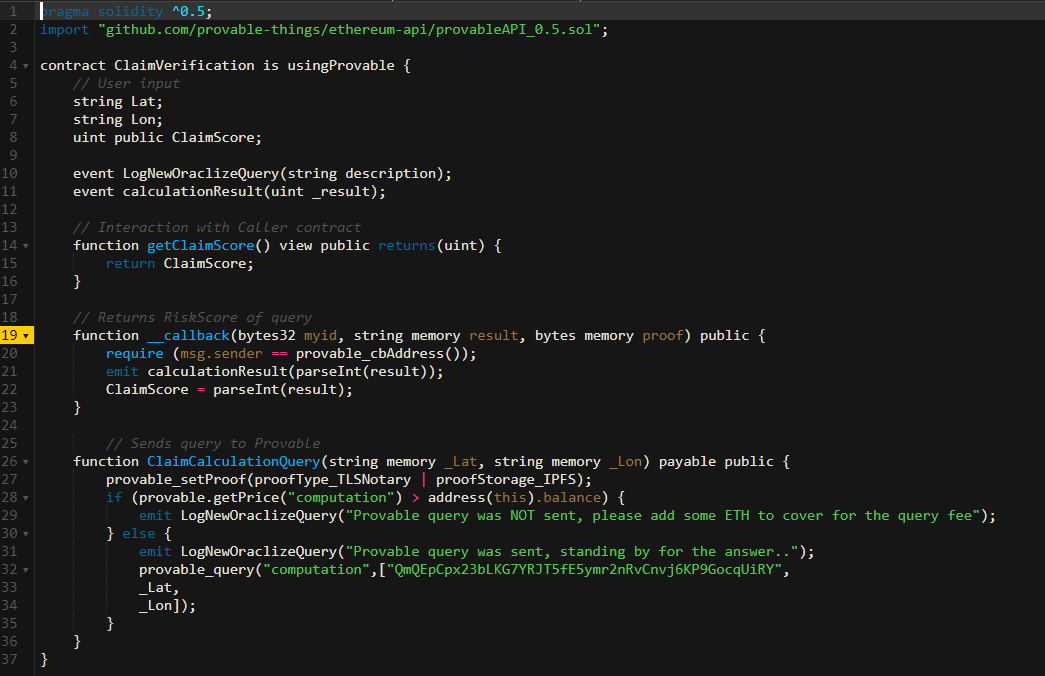
## Appendix 7: Claim.py



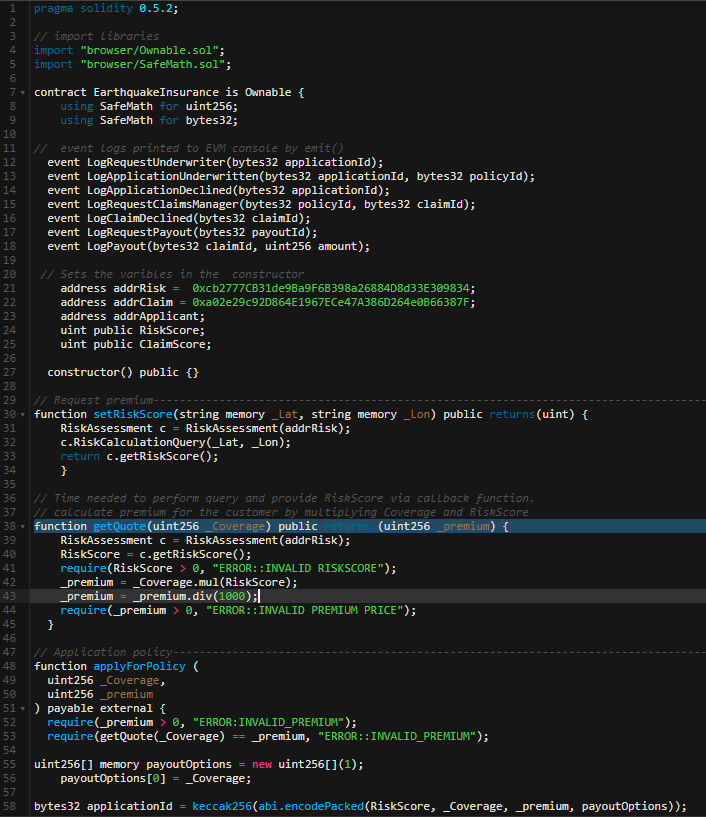
## Appendix 8: RiskAssessment.sol

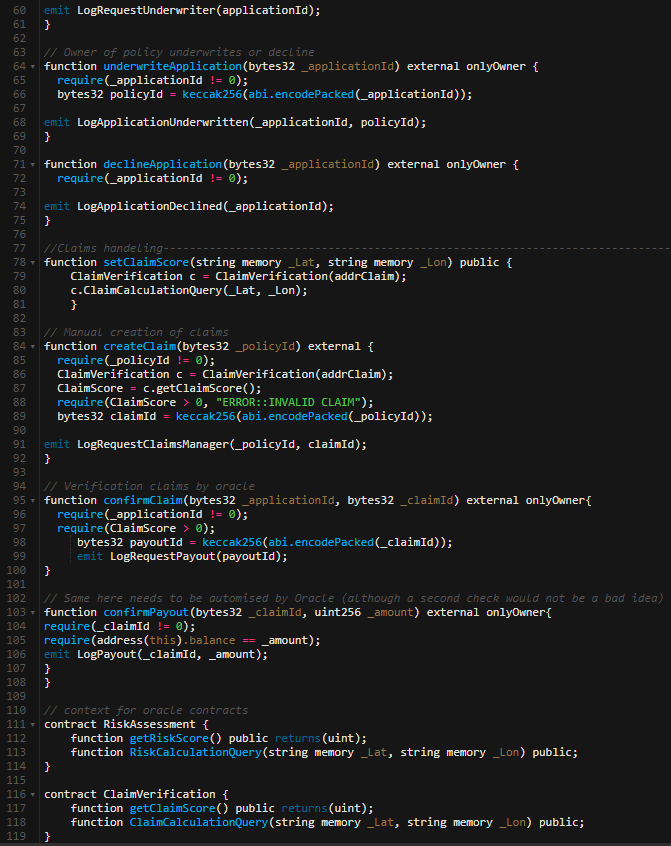


## Appendix 9: ClaimVerification.sol



## Appendix 10: EarthquakeInsurance.sol





## Appendix 11: ImplementationEtherisc.sol

