Supporting Collaborative Care of Elderly through a Reward System based on Distributed Ledger Technologies

Emilien Bai and Kåre Synnes
Department of Computer Science, Electrical and Space Engineering
Luleå University of Technology
Luleå, Sweden
e-mail: emibai-6@student.ltu.se, unicorn@ltu.se

***Abstract*—This paper discusses supporting collaborative care of elderly through a reward system based on distributed ledger technologies. The design and implementation of such a reward system that connect elderly and volunteers by mutual agreements involve technologies such as smart contracts and blockchains. The work is motivated by the demographic change, where an aging population consequently increases the need for care. This causes a great tension in our society, as care resources become increasingly constrained, both regarding costs and availability of care staff. Much of the daily care of the elderly is today done by family members (spouses, children) and friends, often on a voluntarily basis, which adds to the tension. The core idea of this work is to help broaden the involvement of people in caring for our elderly, enabled by a system for collaborative care. The proposed system benefits from recent advances in distributed ledger technologies, which similarly to digital currencies, are build on the ability for mutual agreements between people who do not know each other. The system also benefits from recent gamification techniques to motivate people to collaborate on a larger scale through performing simple daily tasks. The proposed system benefits from inherent distributed ledger technologies advantages, such as a high level of decentralization, thus a high availability, and strong data consistency. These advantages make it interesting to develop the possible links between blockchains and the outside world to allow for a higher level of automation and distribution of services such as collaborative care. New models for distributed ledger technologies, such as Iota tangles or the Swirld platform, may however scale and perform better than blockchains. These should thus be considered for a full implementation and test of the system. In summary, this paper presents a novel framework and prototype implementation of a reward system supporting collaborative care of elderly, that is based on distributed ledger technologies.**

Keywords-component; Blockchain; Collaborative Care; Gamification.

I. INTRODUCTION

The work in this paper in based on a paper presented at the UBICOMM 2017 conference [1]. The aging population has been identified as a challenge for the future in a Swedish study from 2013 [2]. In May 2012, 18.8% of the total population of Sweden was 65 years old or older. This part of the population is expected to reach 20.5% in 2020 and 25.9% in 2060. The main difficulties identified are to finance welfare of the aging population, as well as meeting the increasing demand of service provision. The demand on staff is expected to increase by 210 000 caregivers by 2030 in Sweden, while the supply is expected to stay quite the same. Also, this situation will probably result in a widened financial gap between the cost of welfare and state revenues. The trend of an aging population is confirmed to be worldwide by a United Nations report from 2015, which focuses on the oldest persons (aged 80 years or more) [3].

Much of the daily care (such as performing daily tasks like shopping for groceries, cleaning, cooking, etc) are often performed by informal carers such as family members, or friends. The burden this places on spouses and children of the elderly can often be very high, reducing the quality of life not only for the elderly being cared for but also for these informal carers.

It is thus clear that a broader engagement of our society in caring for our elderly is needed, where voluntary contributions also can be rewarded (besides the altruistic satisfaction of being helpful, pro-bono). Not everyone would of course require such rewards, but motivating a larger cohort of our fellow people may require both short and long term perceived benefits. Examples of short term benefits may be making people's contributions visible in the society or being able to trade work, and long term benefits may include being able to get help back in kind (If I help now, then I will get help later). This leads to the following research question:

How can a system for collaborative care of elderly be designed and implemented to engage and motivate people to contribute with daily tasks on a voluntary basis?

The aim of this work is thus to develop an application intended to connect the population who may need help in common daily tasks with people who may provide voluntarily help. The aim is not to replace workers specialized in health care, but to reduce their work charge where it is possible and therefore instead leave them more time to do important and skilled tasks for the elderly.

Ultimately, by reducing the proportion of paid care, the application may also contribute to decreasing the cost of care for the aging population, without degrading the quality of care.

The rest of the paper is organized as follows. In Section II, we present a state of the art concerning distributed ledger technologies (DLT), blockchains as well as smart contracts. In Section III, we introduce the methodology used in order to develop the system. Section IV focuses on the implementation and design of the system. In Section V, we present the design of the gamification aspect. In Section VI, we discuss how the designed system fills the needs of our research question and point out some limitations. Finally, we conclude this paper in Section VII.

II. STATE-OF-THE-ART

The rapid digitization of our society is key to alleviating the tension on our care systems, where recent technological and methodological advances bring great potentials to enable an increasingly collaborative care. One example is communication technologies, where access to mobile computing now is nearly ubiquitous and where we now at any time can engage in our social networks. A recent example is DLT, including the notions of Blockchains and Smart Contracts, which are introduced in this section together with novel methodologies for user engagement, namely Gamification.

A. Distributed Ledger Technologies

A distributed ledger (also called a shared ledger) is a consensus of replicated, shared and synchronized digital data geographically spread across multiple nodes (sites, countries or institutions) [4]. There is no central administrator or centralized data storage. Instead, a peer-to-peer network is required together with consensus algorithms to ensure that replication and consistency is maintained across the nodes of the distributed ledger.

The most popular distributed ledgers are based on public or private blockchains, which employ a chain of blocks to provide secure and valid achievement of distributed consensus. The first Blockchain was conceptualized in 2008 by Satochi Nakamoto [5] and implemented in 2009 for the digital currency Bitcoin. The example of bitcoin demonstrates the huge potential of blockchains for mutual agreements between two parties without the need of a trusted third party. For example, the volume of daily bitcoin transactions has been over 175 000 since April 2016 [6]. However, the bitcoin blockchain only scratches the surface of the potential of the technology, as it is focused and dedicated on the exchange of value, in the form of bitcoin transactions.

Distributed ledger technologies are expected to have a disruptive effect in our society, especially concerning mutual agreements, as they show many advantages: 1) agreements made on top of the blockchain do not need a trusted third party, and 2) each transaction needs to be signed by its sender using asymmetric encryption, which removes the need of an authentication layer in applications as this is directly handled at the blockchain level. It could ease the exchange of property between people or allow a more fine-grained digital right management.

B. Blockchains

Blockchains are distributed databases for transaction processing, and they are well suited for financial transactions but not limited to such applications. The use of blockchain technology also extends to non-financial applications and is for example, considered for supply chains, asset management or electronic health records.

All transactions are stored in a single ledger and ordered by time. The ledger represents the current state of the system and is replicated across every node. The transactions are broadcasted to the network and accepted if valid, by distributed consensus mechanisms, and are then grouped into a block, which is to be added to the blockchain. The last, and key, operation is to compute an ID for this block before storing it on the blockchain.

This operation can be done by solving a mathematical problem (usually random with a low probability), based on the previous block index (this takes around 10 minutes for the Bitcoin blockchain). The problem consists in finding a nonce (an integer value) to associate with the hash of the content of the block and the id of the preceding block. Once these 3 values concatenated, the resulting hash of this concatenation must respect a constraint: being less than x, x evolving in order to keep a relatively constant period between each block. This constraint can only be fulfilled by trying new solutions for the nonce. Once an ID has been computed, the network adopts the block and begins to work on finding the ID of the next block. The process of computing an ID in this way is called **proof-of-work** and it makes the blockchain immutable since changing an existing block requires to compute the ID for all the following blocks while the blockchain continues to grow. One of the weakness in the proof-of-work mechanism is the 51% attack. In the case an organization controls more than 50% of the computing power, it can start censuring transactions an can refute mining outside of the organization, as the blockchain considered as valid is the one replicated on the majority of blocks, in order to centralize all the rewards.

This operation can also be done using a **proof-of-stake**, where the miner is chosen in a deterministic way. One of the proof of stake design, used in Peercoin [7], is based on the concept of "coin-age". The coin age is a number, which depends on the product of coins times the duration they have been held by the node. The higher the coin age is, the bigger the chances to be elected for the associated node are. Once a node has been selected to mine a block, the duration it held the coins is reset in order to avoid the richest and oldest nodes from dominating the blockchain. This method is more energy efficient than a proof-of-work as it does not imply any power competition between the nodes. It is also safer against attacks as acquiring the majority of the coins is usually more costly than collecting 51% of the computing power of the network.

As the technology evolves, new consensus mechanisms appear. We can cite Proof-of-Elapsed-Time (PoET) [8], used in Hyperledger Sawtooth [9]. This consensus mechanism is based on trusted function called at the central processing unit (CPU) level. It reproduces a leader election protocol found in

many consensus mechanisms and distributes leadership across the population of validators. This mechanism has a low cost of implication, which increases the potential number of validators and therefore the robustness of this consensus algorithm.

Every transaction on a blockchain is signed, using asymmetric key cryptography, ensuring its provenance. The nodes in the network check the transaction conformity (an user can only spend the money he owns from previous transactions and can only perform a transaction in his own name: this is verified thanks to the cryptographic signature), authorization (an user can only perform a transaction in his own name) and resistance to censorship.

If a transaction conforms to the protocol, it will be added to the ledger without any party being able to discard it. All transactions can be processed peer-to-peer without the need of a trusted third party, since a blockchain network does not rely on any central authority but on a distributed consensus.

C. Public and Private Blockchains

This study began with a review of existing blockchain technologies, which revealed two main categories of ledgers: public or private.

A public blockchain is a ledger for which anyone executes transactions or mines blocks. Since anyone can modify a public blockchain, they offer a high replication rate. This is also what makes public blockchains slow and less energy-efficient. Since anybody can contribute to public blockchains, it also offers pseudonymity where an user is only identified by an address and all the transactions referring to this address can be read. Some projects are being developed in order to create a true anonymity when using a Blockchain technology. For example, that is the goal of Zcash blockchain [10] that protect the privacy of its users using zero-knowledge privacy.

A private blockchain does not allow everyone to join. It usually belongs to a single company, or a group of companies, running the chain and validating transactions. It usually uses a certificate authority in order to control the access and the rights of each stakeholder. The level of decentralization is not as good as in public blockchains, but performance is generally significantly higher. Indeed, when located on a public blockchain, a decentralized application represents only a small proportion of the entire system instead of representing the majority of it and, therefore, gains efficiency. It allows for greater privacy since users are chosen and known. However, private blockchains are therefore not resistant to censorship. The main entity running the blockchain can decide to stop one stakeholder to execute transaction.

D. Permissioned and Non-permissioned Blockchains

Our study also identified two subcategories of blockchains: permissioned or non-permissioned. In a permissioned blockchain, each node has a limited role. It may only be allowed to validate transactions, mine new blocks, execute smart contracts (see below) on the blockchain or perform transactions with the chain assets. On the contrary, a non-permissioned blockchain allows any node to take any role. Table I illustrates the resulting categorization of studied blockchains.

TABLE I. CHAIN CLASSIFICATION

	Non-permissioned	**Permissioned**
Public	Ethereum[11], Bitcoin[5], Iroha[12]	Ripple[13]
Private		Fabric[14], Burrow[15], Openchain[16], Multichain[17]

E. Smart Contracts

In 1994, Nick Szabo defined smart contract as [18]:

"A smart contract is a computerized transaction protocol that executes the terms of a contract. The general objectives are to satisfy common contractual conditions (such as payment terms, liens, confidentiality, and even enforcement), minimize exceptions both malicious and accidental, and minimize the need for trusted intermediaries. Related economic goals include lowering fraud loss, arbitrations and enforcement costs, and other transaction costs."

Smart contracts are a way to enforce a legal agreement without the need of a trusted third party. It consists of computer code, stored on the blockchain and its execution can change the state of the blockchain. A sample of smart contract code demonstrating a simple use case of an asset holder contract is illustrated in Figure 1. It profits from blockchain immutability to ensure that the terms cannot be modified. Thus, smart contracts cannot be modified and the result of an interaction is predictable and not corruptible. A high level in data integrity (as well as a good log level in case of breach in contracts design) is therefore ensured. Smart contracts develop the need of experts able to formalize legal agreements and convert it in clear and complete specifications. These specifications have to be translated in computer code and audited in order to ensure that all corner cases are covered.

```
pragma solidity ^0.4.11;
contract Bank {
    // We want an owner that is allowed to selfdestruct.
    address owner;
    mapping (address => uint) balances;
    // Constructor
    function Bank() public {
        owner = msg.sender;
    }
    // This will take the value of the transaction and add to the senders account.
    function deposit() public payable {
        balances[msg.sender] += msg.value;
    }
    // Attempt to withdraw the given 'amount' of Ether from the account.
    function withdraw(uint amount) public payable {
        // Skip if someone tries to withdraw 0 or if they don't have enough
        // Ether to make the withdrawal.
        if (balances[msg.sender] < amount || amount == 0)
            return;
        balances[msg.sender] -= amount;
        msg.sender.transfer(amount);
    }
    function remove() public {
        if (msg.sender == owner){
            selfdestruct(owner);
        }
    }
}
```

Figure 1: Sample of smart contract code

F. Bitcoin and Etherum Smart Contracts

The Bitcoin blockchain only runs a single smart contract, where the Bitcoin blockchain only ensures that the sender actually owns the tokens he wants to send in a transaction. As a consequence, a receiver cannot refuse a transaction. It is usually not referred to as a smart contract since the code is more embedded inside the chain protocol than actually running on a chain virtual machine (VM).

Ethereum is the biggest public platform for smart contracts [11] which provide a Turing complete language for smart contracts executing in a virtual machine environment. As it is a public and uncensored platform, all users are free to send their own code, to be executed by the Ethereum VM. To avoid malicious user from locking the system, executing infinite loops algorithm for example, the VM use a "gas" system. When sending code to be run by the VM, users have to send a certain amount of gas associated with it. For every computation cycle required by the contract execution, a small amount of gas is consumed. If there is no more gas to consume, the computation is stopped, and an error is returned. Gas is bought using the chain currency (in our case ether): Its consumption is used to reward the nodes who took part in the smart contract execution. Smart contracts are triggered by receiving a transaction and they process transaction data in order to change the state of the contract. The code execution is replicated on each mining node in order to validate the transaction and include it in the next block. Code execution happens as many times as there are nodes validating transactions. Therefore, there are some limitations in the smart contracts design [19]: it is difficult to link smart contracts with outside world events, or make use of external services automatically (without a transaction). If a contract is waiting for data from the outside world and it does not receive the same data on every node, this would create a conflict on the chain. That is why smart contract are mainly triggered by transactions that ensure data consistency across the nodes. However, one library, Oraclize is aimed at bridging external service with smart contract code in a secure way, but has not been tested in the field of this study. On the contrary, if a contract must call an external Application Programming Interface (API), which will trigger an action, it cannot determine which node is responsible to actually make the call. This design issue can be avoided using external logs watcher that can check a contract status then trigger the outside chain action if needed.

Smart contracts are thus not well suited to ensure agreements outside the chain, but they remain a very efficient way to condition fund transfer inside a chain. We can imagine an internal coin system with which users agree on a value and use it to limit the volume of "official" currency. This mirrors the current financial system where major stakeholders like banks agree on the value of a debt toward each other and balance the debt volume without actually exchanging assets. Finally, smart contract are also not well suited to hide confidential data, especially on a public chain. Every node replicates the database, and can try to brute-force encryption of data if need be. Also, every transaction is relatively anonym, which means activity of user towards contracts can be traced.

G. Involver

Our technical review only identified one mobile application for volunteering, an application called "Involver". It is defined as a social volunteering platform [20].

The goal of this application is to bring together potential volunteers with partner organizations that need help. Every cause a volunteer can help with is ordered based on location, subjects and skills needed. The rewards are brought by sponsors and take the form of non-monetary advantages.

The application also offers to certify the number of volunteering hours on professional social networks. It also includes a social aspect emphasizing the fact that volunteering is more interesting with friends. This example illustrates the need of a trusted third party (in this case the application) when agreements are made between volunteers and organizations.

Involver is however more of a start-up than a scientific platform. It is also not aimed at manipulating sensitive data, as this kind of information is to be held out of the application, by the organizations themselves if need be.

III. METHODOLOGY

This work is based on multiple theories within the gamification field, which form the basis of the theoretical work as well as the implementation. This section describes the definitions in the context of this work.

A. Gamification Definitions

The word gamification appeared for the first time around 2002, when Nick Pelling used it for its consultancy business [21]. Gamification is according to Hutoari and Hamari [22]*"a process of enhancing a service with affordances for gameful experiences in order to support user's overall value creation."*

Deterding et al. [23] propose a more general definition of gamification as "*the use of game design elements in non-game contexts"*. This definition is supported by the distinction made between games and play, with gaming being more structured by rules and more competitive. Game elements are defined as elements that are characteristics to games, found in most (but not necessarily all) games and found to play a significant role in gameplay.

Since gamification has been a trending topic, it prompted a lot of academic studies, which showed gamification to be present in many different contexts such as learning (e.g., Duolingo [24]), exercise (e.g., Fitocracy [25]), work and more.

B. Gamification, Rewards and Volunteers

The gamification aspect is part in the final application as an incentive for volunteers to use it. A review of studies concerning gamification by Hamari, Koivisto, and Sarsa [26] show that, globally, gamification has positive effects and benefits on users where it is used. Gamification have a

positive impact on the behavior of users, but also a psychological impact, acting on motivation, attitude or enjoyment of users while filling tasks. The gamification aspect is expected to motivate users and maintain their involvement.

Gamification can also be coupled with rewards in order to extend the scope of the gamification to the real world. A reward system can thus be seen as a natural part of a gamification system. For example, it can be used as a mechanism to engage, motivate and compensate users who volunteer their time and services for collective purposes.

Volunteers' motivations to help have been shown to be more based on intrinsic rewards (to fulfill psychological needs). However, small and non-expensive rewards are also appreciated and encouraging, where, generally, the goal is not to spend as much in a reward as the cost to pay someone [27]. The vision of combining gamification with real-world rewards allows reaching both fully altruistic people, who probably would not use the rewards or would not see it as an essential part of the application, as well as an audience needing more recognition to maintain its motivation Rewards are also expected to stimulate the interest of people usually that do not usually take part in volunteering activities.

C. Achievements and Badges

Achievements are a really common part in gamified applications. They are usually associated with badges and find their origin in merit-badges given to boy scouts of America since 1911. In 2003, Wikipedia started Wikipedia's Barnstars [28], aimed at rewarding contributors for their involvement on the platform. Another example of successful use of achievements is the Foursquare badges [29], which encouraged people to complete tasks in real life in order to unlock them. Furthermore, all games published on the Microsoft Xbox Live [30] platform are required to have achievements. A study by Anderson et al. [31] showed that badge placement in an application can have an effective influence on user behavior and also affect his/her use of the application. However, a study by Montola et al. [32] concludes that achievements globally have a positive effect on motivation but can sometimes be confusing for some users if they are not introduced properly. In summary, badges can be efficient incentives and are relatively cheap to implement in an application.

D. The Hamari and Eranti Achievement Framework

According to the framework designed by Hamari and Eranti [33], an achievement can be divided in three main parts.

Firstly, an achievement has a signifier, which is the visible part of an achievement and conveys information about it. It consists of a name that set the theme of the achievement and hints at the completion logic for it. The signifier also includes a visual, which completes the name and often has two states, unlocked where the visual is faded and completed where the visual gets fully colored. Finally, the signifier has a description, which describes what is required from the user to complete the achievement and what can be gained by completing it.

Secondly, an achievement also consists of a completion logic. It consists of a trigger, a pre-requirement (specific date, already completed achievement), a conditional requirement to determine if the action is triggered and also a multiplier, which determines how many times the three first parts have to be completed to unlock the achievement.

Thirdly, achievements carry rewards to show the user the achievement that has been completed. When added to a game, achievements completion can be a way to unlock in-game rewards. The external part of the reward is often the fact that these achievements are displayed publicly.

E. Leveling

Leveling based on experience points is an easy way for users to keep track of their progress. It was originally used in role playing games, and then extended to any type of games. The logic behind leveling is quite simple: when performing a task, users receive points, and then when a certain amount is reached, the user advances a level. In games, earning levels is often linked to gain or progress skills for the avatar. In a gamified application, advancing a level is recognition of the skills acquired by the user in real life: it can also allow an user to access more advanced features. In games, points are earned when completing a mission/quest: in gamified applications, points are delivered when the user completes the task the app is trying to help with. For example, points can be delivered when a volunteer completes an offer, based on the number of tokens earned. However, when the user spends his tokens, he keeps the same number of points.

An important part of a levelling system is the threshold: it represents the number of points needed to reach the next level. Usually, the first levels have a low threshold, in order to keep the user motivated and show quick progress. Then, once users have been significantly engaged, thresholds get bigger to be more challenging and therefore more rewarding.

IV. IMPLEMENTATION

Based on the pros and cons listed above, the Burrow blockchain has been chosen to conduct our test implementation. It is fast, provides a smart contract virtual machine and the permission layer allows controlling the access rights. As the system works with sensitive data, it benefits from the privacy a private blockchain provides. The permission layer allows limiting the number of nodes allowed to mine blocks or create contracts on the chain. Our chain is therefore only dedicated to our system and does not spoil resources for other contracts. It also uses a proof-of-stake consensus mechanism, which is better suited to the use of a private blockchain, since every mining node is known and trustworthy. A proof-of-stake consensus is also more energy efficient and faster than using proof-of-work.

A. System Goals

The designed system is intended to connect elderly with volunteers who can help them with everyday life tasks, which do not require any specialized skills (for example, in health care). The goal is not to replace health care workers but to reduce their workload where it is possible in order to give them more time for specialized tasks. As an incentive,

volunteers receive a non-monetary reward, a token based on time spent to help and eventually resources involved in tasks, as the use of a vehicle for example. These tokens can be used to acquire rewards as coupons or advantages / discounts in local shops or non-monetary advantages. On top of this, gamification aspect using points and badges is added to keep volunteers motivated.

Users of the platform need to register by giving their identity. Their personal data is stored encrypted and only revealed to other users if they share a task (tasks can also be referred to as offers or task offers from elderly users). Once users are registered, an authority is charged of verifying the information and grants the permissions according to the user status. For example, this external authority has to check that volunteers who claim to have a driving license actually have it, and more generally verify the identity of users who register. This authority could follow the model used by the car sharing application for example. An elderly user is allowed to create task offers: they describe the mission, specify a time slot when it has to be performed and duration, and a type for the task (gardening, shopping, accompanying for visits, etc.). The reward amount is computed according to the offer specifications. A task that requires the volunteer to own and use a vehicle will be rewarded with a higher reward then task with no material need. Once the offer is created, volunteer users can see it and read its specifications: if a volunteer is available and able to fulfill the task, he or she can commit to it. From there, the elderly user can access the volunteer's contact information to schedule the task more precisely. Once the task is accomplished, the volunteer needs to claim the reward. The elderly user can then confirm that the offer has been fulfilled: this action triggers the issuing of tokens for the volunteer who helped. With these tokens, the volunteer will be able to buy rewards. Rewards are added by rewarder users. These rewards contain a description, a price and a code, delivered only when the reward is bought. This is illustrated in the activity diagram below, see Figure 2.

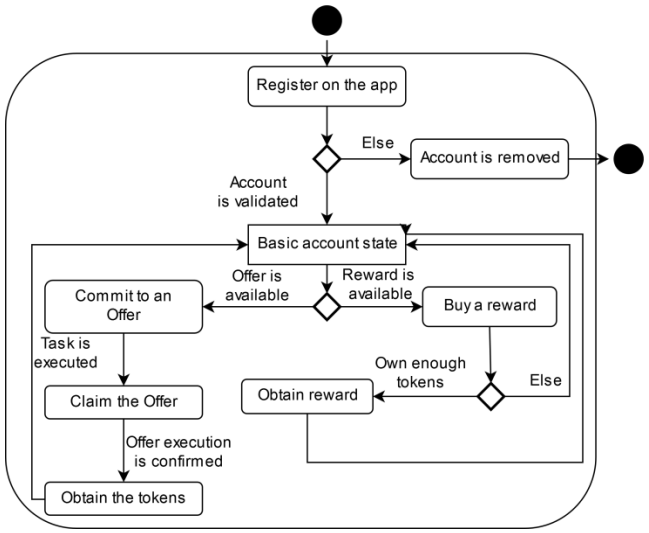

Figure 2: Activity Diagram for Volunteer Users

B. Global Design

The system back-end is built on top of a blockchain with smart contracts to handle agreements between the users. It has four main contracts handling the different parts of the system: these contracts form a database while they also ensure system consistency. These database contracts do not directly store data. They are more data structures that reference other contracts where the data actually is.

The bank contract handles the tokens for each user: the only way tokens are issued is when an elderly user confirms that a task offer has been fulfilled. The only way to use these tokens is when a volunteer user spends them to buy a reward. The bank contract stores the balance for each user and ensures that the user actually owns enough of them before spending them.

The user contract is used to store user data. One part of this data is readable by everyone (and uses pseudonimity) while sensitive data stays encrypted and is only revealed when a task links two users. This contract is also used to handle permissions for each user. Permissions are set by an authority according to the status of the user: depending of his permission level, an user can or cannot perform some actions in the application (As an example, only a volunteer user can commit and claim an offer).

A contract is used to store offers and commit, claim and confirm their execution. An offer is a task, proposed by an elderly user for which help from a volunteer is needed. Thus, offers are smart contracts with properties and states: the state of the offer evolves during the course of the agreements but properties are immutable. This evolution is described in Figure 3.

Finally, a contract is used to store and buy available rewards. Rewards are added by partner rewarders in a limited availability and bought by volunteers.

The blockchain handles authentication of users for these actions, allowing a mutual agreement between different users without the need of a trusted third party, once the registration is complete. The blockchain also guarantees the content of agreements since contracts cannot be discreetly modified. The division of the application is described in Figure 4.

C. Detailed Architecture

The system is built on the Monax blockchain, Burrow [15], a fork of the Ethereum blockchain allowing working with a permissioned ledger: this permission layer also allows using a proof-of-stake mining mechanism. Another difference compared to unpermissioned ledger is that nodes can have restrictions on how they can contribute: some can be dedicated to validate nodes, while others handle permissions or receive transactions.

Contracts are developed using Solidity [34], an object-oriented programming language for smart contract development. Solidity code of a contract needs to be compiled outside of the blockchain, and then is sent using a specific type of transaction. The result of this operation, if successful, is the address of the contract. This address will then be used to interact with the contract, by calling its functions in transactions or when reading its state.

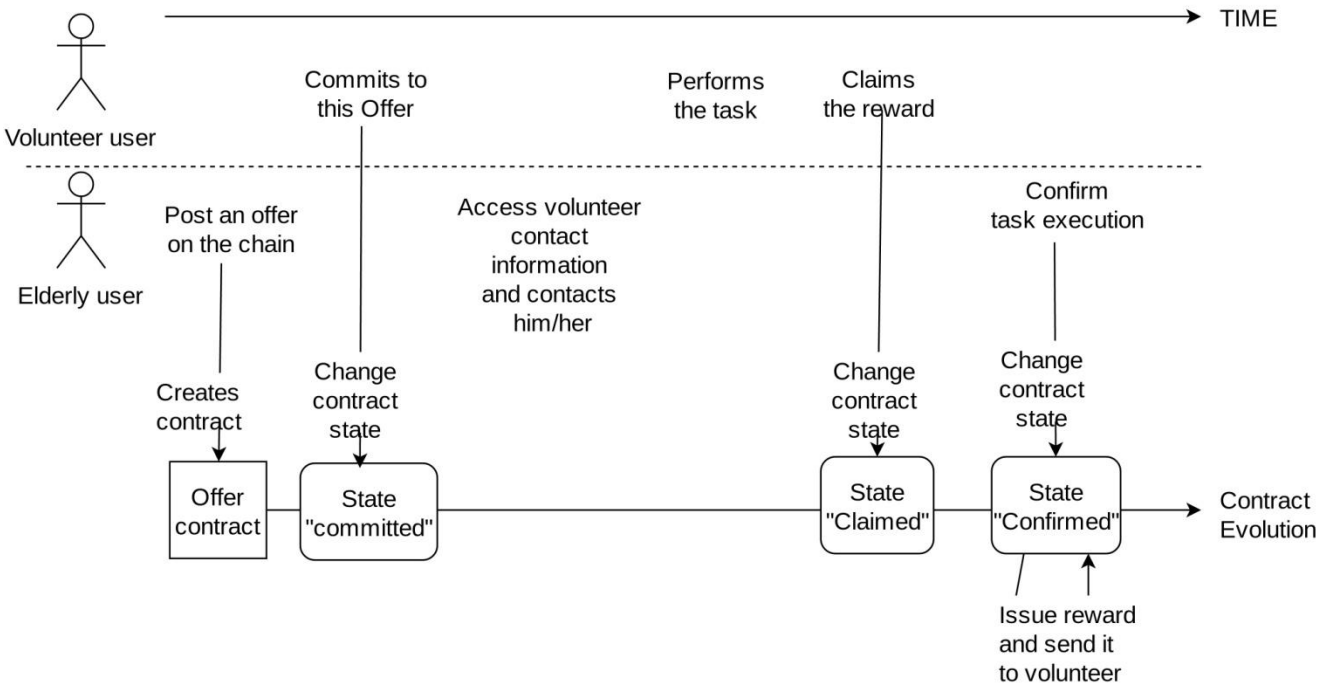

Figure 3: Offer evolution

The application is developed following an action-driven architecture coupled with a five types model.

The five types model suggests splitting the application using different kind of contracts. Database contracts, where data is stored, can be read or updated. As explained before, these databases are key value store for our data. They consist in lists where a contract address can be found using its id. The use of a custom data structure (our lists) over the one usually used when developing in Solidity (mappings that are dictionary already existing in the basic structures of Solidity) has been done in order to be able to query all the entries in a database, when we would have need to keep every data contract names to retrieve them if using the mapping structure. Controllers contracts operate on database contracts, and can operate on multiple databases (for example, read user's permissions from one database and then operate an action on another). A third type of contract is contracts managing contracts (CMCs) where other contracts addresses are kept in view and can be replaced if needed (if we update the code of a controller for example). They provide single point of entry to the system, which is useful when a system uses many contracts and therefore also many addresses. They include an update mechanism for controller contracts in order to be able to edit a code that would otherwise be immutable. These CMCs allow the update of existing controller contract in a transparent way for the user. Without them, when updating a controller contract, the user would need to obtain the address of the updated version of the contract. Using this system, the user only need to query the contract wanted using its name and the CMC directly redirect the query to the latest version. Application logic contracts (ALC) are contracts specific to an application and they perform multiple operations using controllers and other contracts. Finally, utility contracts can be seen as libraries: they perform a specific task, without modifying the state of other contracts and can be used without any restrictions. They are not project specific and can be re-used in different situations.

The five types model effectively separate actions used to interact with databases contracts. Actions are thus focused on small parts and modifications of the system. Actions are smart contracts with only one function (in our case, "execute") and perform atomic modifications to the system. It can be seen as a microservice architecture even though it does not share the goals of such an architecture but appears more as a need to be able to maintain and update the application. Actions are stored in a CMC. This architecture allows the system to be updated more easily than if using full controller contracts. A full controller would handle every interaction with database contracts. Any simple modification to a simple function implies the full contract replacement, which infers heavy interaction with the chain. As a result, actions can be dynamically replaced without the need of modifying a complete controller contract. As we can see in Figure 4, the update mechanism is integrated directly in the action driven architecture. Users with the right permission level can add, replace or remove an action from the system in the same way more general users interact with it. The main CMC is a Decentralized Organization Upgrade Guy (DOUG) storing all databases' contracts of the system and especially, the action database. The action database is also a CMC and works with an action manager calling this database in order to find the actions to execute. By following this architecture, an user can interact with the system knowing only the address of the DOUG and the databases public APIs. On the developer side, maintenance operations are

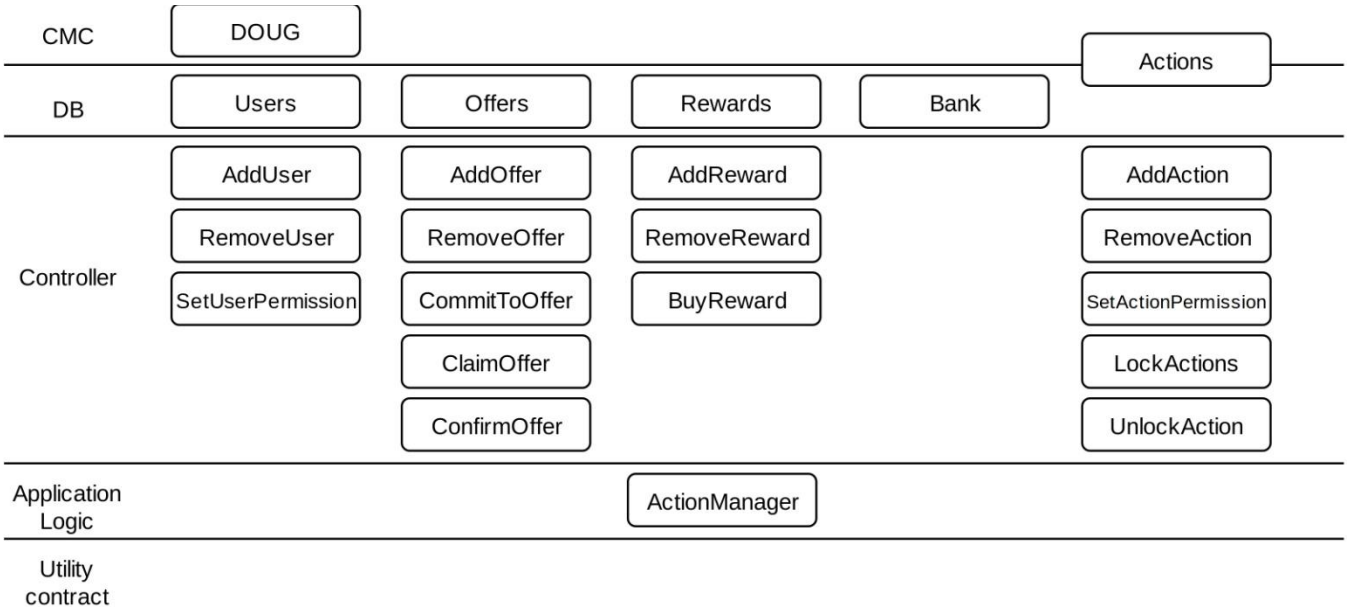

Figure 4: The Five Types Model

simple because actions can be easily updated (replacing an action in the database is an action itself) without the need of updating every interface as long as the DOUG contract stays the same (and therefore keeps the same address). Since action replacement is an integrated part of the application, this also allows using multiple developer accounts with the right permission level instead of locking the system by limiting updates to only its creator. The process of executing an action is described in Figure 5. First, users, that only know the address of the DOUG, query this contract in order to get the address of the action manager (1). Then, users then send a transaction to the action manager address using as parameters the action they want to execute (2). The action manager contract receives the transaction. It queries the DOUG in order to order to get the address of the action database (3). The action manager then queries the address of the action contract that will be executed (4). The action manager repeats step 3 to get the address of the user database (5), and repeat action 4 in order to get the calling user data and especially permission level of the user (6). From the data, it can read in the action contract, the action manager verifies that the caller user has the right permission level to execute this action (7). Then, it calls the execute function of the action contract if allowed to do so (8). The action contracts query the DOUG to get the addresses it needs to perform its action (9), and modifies the database data accordingly (10). Databases are locked in such a way that they can only be modified by the action being currently executed. The action returns the result of its execution to the action manager (11) that returns it as a result of the transaction originally sent by the user (12).

D. Interface

In order to keep the system as decentralized as possible, the user credentials are not kept in a database. The most suitable solution is to let users manage their own credentials and this can be done using a mobile or desktop application acting as a Bitcoin wallet. This solution presents as an inconvenient a high risk of credential loss. It should only be coupled with an efficient "save-and-restore" system that would allow users to keep a safe copy of their own. Another solution to store credentials can be paper wallets: these are cardboard-cards with flash-codes or text-written credentials. This can be a solution to effectively handle permissions and verify user information by sending them their credentials using for example, standard post, or asking them to present themselves to an office where the identity verification occurs and their paper wallet is delivered. The credentials on paper-wallets can afterwards be stored in a mobile application or required to be scanned for every action performed through the application. This solution has not been chosen in the prototyping step, but should be considered in a following step.

The user then interacts with the blockchain through a Representational State Transfer (REST) API. Every node used as a validator for the blockchain is also used to host an API server, allowing a good level of decentralization. The choice of using a REST API comes from technical limitation at the moment the project has been conducted. An ideal design applies the creation, signing and sending of the transactions directly from the device it is sent, and not centrally executed on a third part server. The use of an installed application instead of a web based application also limits the number of request needed for developing the gamification aspects, since, as a Bitcoin wallet, this kind of application does not need to store every transaction but only those concerning users. The developed application used for testing is illustrated in Figure 6.

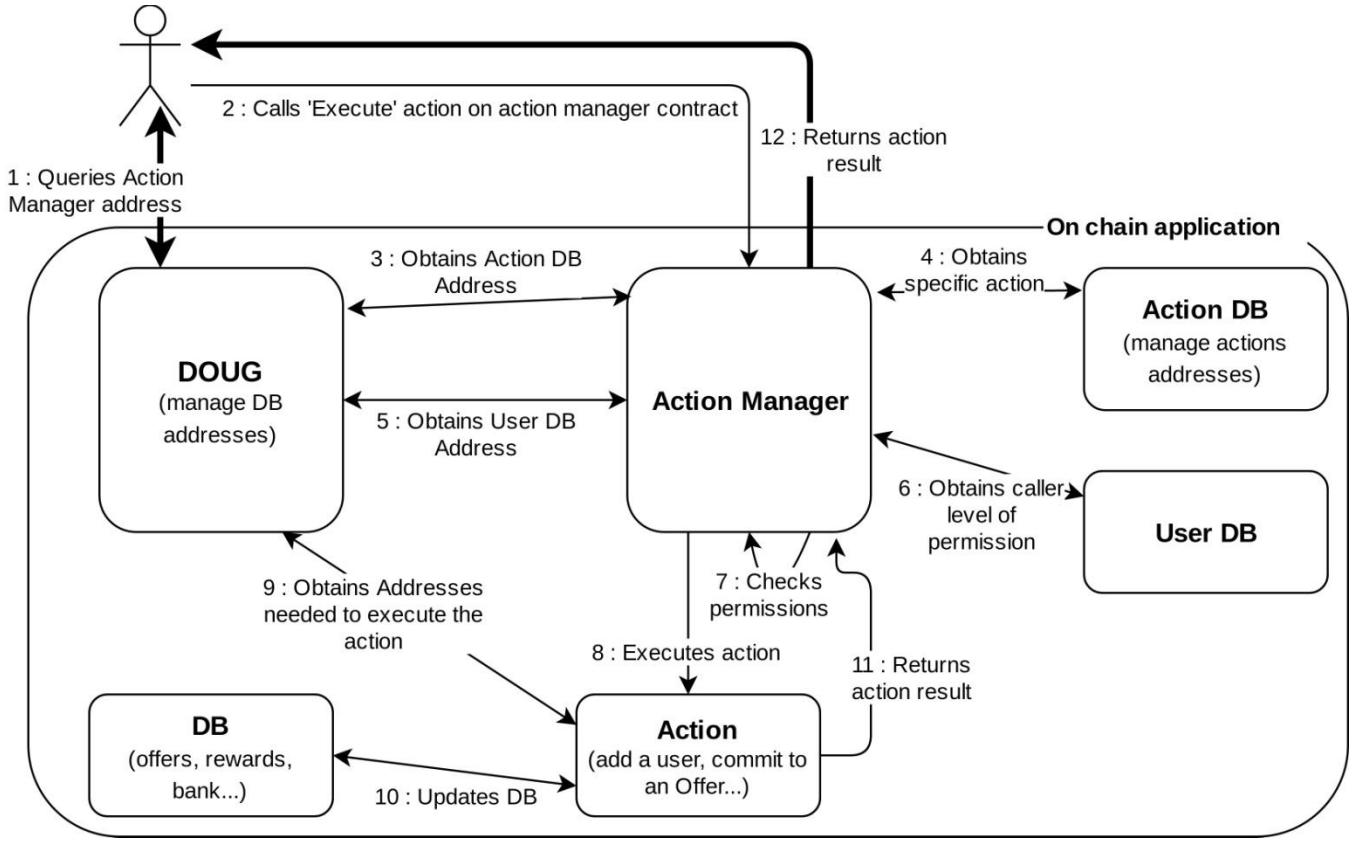

Figure 5: Execute Action Process

E. Technical limitations

The system was first designed to be as modular as possible: every action had its own execute method, taking various numbers of arguments. Since this modularity implies no real inheritance, the action manager had to use low-level calls, a Solidity feature where a function is called at a specified address without knowing the contract API at the caller level. These low-level calls return a boolean only indicating if the call succeeded (a function has been found and called) or not, but not the actual return value. This solution has been abandoned since it created a lot of problems in data formatting for arguments at the action manager level and afterwards at the action level. The absence of a relevant return value was also a performance issue since it implied to check every action execution afterward. Finally, the choice has been made to use a formatted schema for the execute method of every action, covering all the current cases, and ignoring some useless parameters for some of the actions contracts. This choice reduces genericity but improve the reliability and performances of the system.

Some gamification aspects have been limited by the use of smart contracts as databases. This layout is not really efficient when querying many contracts and therefore limits some features such as ranking between all users. This limitation is also linked to our choice of using list as our main data structure. This choice is easy to deploy and reliable but does not scale really well or not really well suited to filter elements.

V. GAMIFICATION DESIGN

The choice for our application is to limit the gamification aspect to the frontend in order to reduce the volume of interaction with the back-end and therefore improve performance. It has been a design choice from the beginning, as there were no existing studies concerning performance and scalabity of decentralized applications, either on public or private blockchains. Badges fit really well with this vision. Achievements are well oriented towards volunteers, as they are the target we try to motivate. The achievements implemented have two main objectives. The first objective is to serve as a tutorial, where these achievements appears when doing really basic actions (such as to commit to an offer or getting a validated account) and are supposed to show the possibilities of the application while also introduce the achievement system. This kind of rewards is supposed to be numerous in the beginning: it guides the users toward using all the features of the application and rewards them quite often in order to provoke a feeling of significant progress and create engagement. The second objective is to maintain motivation and encourage involvement. The achievements for this objective are focused on quantity and regularity: it consists in fulfilling a defined number of offers,

typed or not: buying rewards, keep checking the app and keep fulfilling offers every week/month to create strikes.

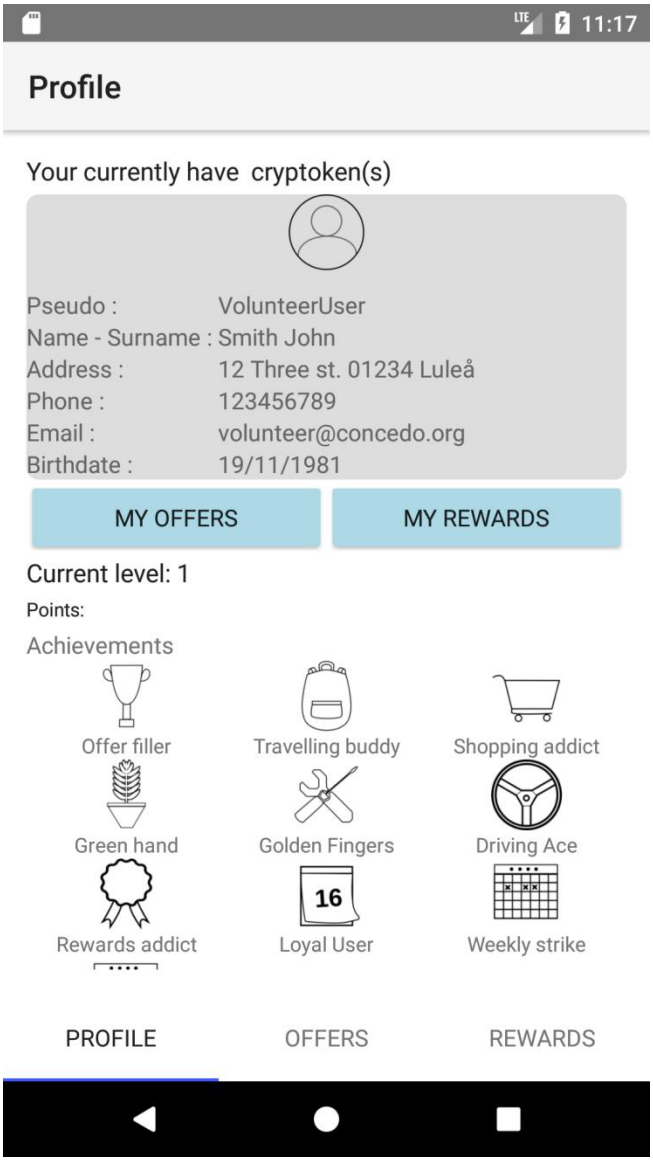

Figure 6: Application Interface for Volunteers

The achievements implemented in the application are detailed in Table II.

TABLE II. LIST OF ACHIEVEMENTS

Achievement / Multiplier	Step 1	Step 2	Step 3
Complete X offers	10	50	100
Buy X rewards	10	50	100
Complete X offers- Gardening	5	20	50
Complete X offers- Shopping	5	20	50
Complete X offers- Driving	5	20	50
Complete X offers- DIY	5	20	50
Complete X offers- Accompanying	5	20	50
Use the app for X days	30	180	360
Complete X offers in a week	2	4	6
Complete X offers in a month	4	8	15

The Hamari and Eranti Achievement Framework, as detailed above have been utilized to design the achievements for the application. Since no formal study have been found regarding leveling curve formulas, and since in game examples are often based on experience points gains varying depending on level, the following formula has been chosen in order to progressively increase the levelling thresholds:

$$t_1 = 30$$
$$t_{n+1} = t_n + \frac{l_n}{5} \quad (1)$$

The chosen initial threshold (to pass from level 1 to level 2) is 30. Since points are approximately equivalent to minutes, this allow users to gain levels quite quickly at first, then require more engagement to level-up further. For example, the users need a bit less than 4 hours of cumulated engagement to reach level 5. Reaching level 10 requires around 13 hours for the volunteer and reaching level 20 is almost 90 hours. Having such a progression is aimed at challenging users who will need to dedicate more and more time in order to level up. The progression of points needed is illustrated in Figure 7.

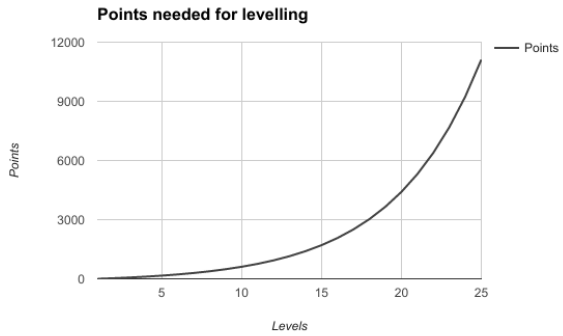

Figure 7: Levelling Curve

Levels can be seen as a simplified achievement since it can be associated with a name and a visual status. The chosen completion logic, focusing more on global progress, is however simplified in comparison to an achievement focused on specific task completion. Gaining points also happens more often than unlocking an achievement.

VI. DISCUSSION

How can a system for collaborative care of elderly be designed and implemented to engage and motivate people to contribute with daily tasks on a voluntary basis?

A system for collaborative care of elderly should come as a complement to the "classical" care system. It means that it should be efficient to help professional workers while keeping the costs low and the elderly population safe. The costs problem can be partially treated by implying volunteers in the process of elderly care and creating a system where

moderation needs are low. In order to keep these cost low, the system needs to be reliable and able to work with a very high level of autonomy. DLT appear to be interesting by their performance in handling mutual agreements between parties not knowing each other. Such a system only requires moderation at registration and then can handle itself efficiently. Such an application should be mobile in order to integrate itself efficiently in daily life for both volunteers and rewarders. This mobile deployment is really compatible and almost comes as a complement for such a distributed system.

Even if the system benefits from the advantages of the blockchain on mutual agreements, it also shows some limitations due to the fact that it requires many interactions with the outside world. Firstly, the designed system can be abused by two parties knowing each other and agreeing on hypothetical tasks in order to issue tokens. This bias can partly be solved by limiting the number of offers an elderly user can post every week/month. The risk is to disadvantage honest users who need a lot of help, while only curbing abuses. However, since every commitment can be publicly visible, these abuses could be detected and user banned although this implies more authority regulation than just certifying user identity at registration. Therefore it reduces the interest in using such a trust-based system.

Another issue comes from the fact that the system does not allow nuances: a task offer will either be confirmed or not. Even if task confirmation could be coupled with a notation system, weighting the reward would be really dependent on personal appreciation. In the worst situation, an offer is not completed at all and this case is not automatically disadvantageous for the abuser while it can have negative consequences for the elderly user. Nevertheless, this situation is the same in every system implying trade between nonprofessional users (such as in carpooling services for example), and require an impartial arbiter to be resolved. However, what differs from another service is the criticality of the failure from one of the user. If a volunteer does not execute a task, it can have critical consequences for the elderly user while the volunteer only faces being banned or some equivalent penalty. It would require a legally recognized contract agreement in order to avoid this kind of situations. Such a protocol could discourage the potential volunteers. It could also create confusion between volunteers and professional workers.

Finally, another bias that could possibly appear is the preference for the most rewarding offers at the expense of the smaller ones. Even if rewards are calculated based on the efforts needed for their fulfillment, the least demanding offers could be discouraging because of the external efforts it can imply.

In this paper, we discuss many use cases where DLT could have a disruptive effect compared to our current applications. The main aspect is the focus on removing the need of a trusted intermediary in different kind of exchanges (monetary, intellectual property, assets management). However, we already have and use solutions daily to tackle this kind of scenario. And problems that come from theses scenario are usually not linked to a lack of trust in the intermediary. In our financial system, the vast majority of population trusts the banks and does not need any kind of censorship resistant money that would require users to engage more time and energy than what we currently have for quite a similar result. Moreover, the crypto-currencies based on blockchains as we design them today are not compatible with the functioning of the financial system based on debt and monetary creation. It would require either a huge adaptation from the financial system to fully embrace crypto currencies based on blockchain (which is highly unlikely) or a transformation in the paradigm we use to design crypto currencies and therefore, would result in the loss of what made a huge part of their interest: being a digital cash. Another interesting feature of smart contracts is the automation level they can provide. Yet, this level of automation is also available in our current systems when based on an external trusted authority. DLT could only enable the deletion of this intermediary but not actually creating new processes.

Not to be fully negative on the subject, we can foresee interesting use cases concerning bookkeeping and trace-ability. DLT may reveal themselves interesting when it comes to store important and non sensible data in an immutable way for an extended amount of time for example. Private blockchains could also be put in use to synchronize swarms of machine working together independently and not requiring a centralized synchronization. But even if the problems we approach today using DLT are “non-problems”, it is still an interesting field of research as we need time to explore the full capabilities of this relatively new technology.

VII. Conclusion

Care of elderly is an important and sensitive topic, which raises many and various concerns. The need for care will grow and volunteering will have to take part in this care in order to maintain reasonable costs for the society, as well as a sufficient level of services. A service to establish contact between people needing help and people willing to volunteer therefore is well motivated. This kind of system creates mutual agreements between users not necessarily knowing each other and can therefore take advantage of DLT.

Smart contracts are most efficient with on-chain agreements but show limitations when interfacing with outside-chain events. Interfacing with such events requires additional control points during the course of the agreements, to keep consensus between users. This reduces the interest in comparison with traditional, often centralized, systems between non-professional users.

In conclusion, the system described here still benefits from inherent DLT advantages, such as a high level of decentralization, thus a high availability, and strong data consistency. These advantages make it interesting to develop the possible links between blockchains and the outside world to allow for a higher level of automation and distribution of services such as collaborative care.

VIII. Future Work

This proof-of-concept system and prototype would be required to be evaluated with real users, first in small scale through participatory design and then in larger scale to

ensure statistical certainty of results. The current project can be seen as a feasibility study to pave the way for this future work that would require more extensive resources.

The next development step in this project would consist in working on a new agreement protocol that could for example, involve a third-party user of the system to settle and confirm or not the task execution. This could be associated with an appreciation system working both ways: the elderly user evaluates the service he received in terms of motivation or punctuality (the goal is not to judge the skills) while volunteers could rate the offer description accuracy and the reception received. By adding a rating system for both tasks and users, it should motivate users to provide a quality service and filter abusive users or at least point them out. Using smart contracts, we can imagine to automatically suspending accounts who received a very bad appreciation in order to clarify the situation with the authority running the service.

We could also think of adding more filters to offers based on the elderly user preferences and needs: for example, some tasks may require a valid driving license that can be authenticated at registration: then, only users with a valid driving license would be able to see and commit to these offers. One last feature that can be investigated is a bidding system allowing volunteers to compete on committing to an offer and, therefore, not base the system on a first-come, first serve model. This would probably allow a selection focused more on the volunteer motivation.

A future step would also be to include more in-life elements, starting with paper wallets in order to authenticate users for the first meeting or while claiming a reward. Offer passwords, needed to claim the offer, shortly evoked in the precedent paragraph taking the form of matrix bar-code only readable by the mobile application could be used as a proof of the meeting while complicating frauds.

Finally, the concepts of utilizing IOTA tangles (iota.org) or the Swirld platform (swirlds.com) to replace blockchains as mechanisms to implement trust and data sharing needs to be further investigated. These could mitigate the inherent problems with many of the popular blockchains used today, such as scalability and performance issues.

IX. ACKNOWLEDGMENTS

This work was sponsored by Vinnova and supported by Ericsson Research in Luleå through the academy-industry exchange project Concedo. The work has previously been published in a shorter paper at UBICOMM 2017 [1].

REFERENCES

[1] E. Bai and K. Synnes, “A Reward System for Collaborative Care of Elderly based on Distributed Ledger Technologies”, In the proceedings of the Eleventh International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM), November 12-16, 2017, Barcelona, Spain , [ed] IARIA, ISSN: 2308-4278, ISBN: 978-1-61208-598-2.

[2] P. M. Office, “Future challenges for Sweden”, 2013, [Online], Available: http://www.regeringen.se/49b6cf/contentassets/389793d478de411fbc83d8f512cb5013/future-challenges-for-sweden--final-report-of-the-commission-on-the-future-of-sweden, [retrieved: 10, 2017].

[3] U. N. D. of Economic and S. A. P. Division, “World population ageing 2015”, 2015, [Online], Available: http://www.un.org/en/development/desa/population/publications/pdf/ageing/WPA2015_Report.pdf, [retrieved: 10, 2017].

[4] UK Government, Office for Science, “Distributed Ledger Technology: beyond blockchain”, 2016, [Online], Available: http://www.ameda.org.eg/files/gs-16-1-distributed-ledger-technology.pdf, [retrieved: 09, 2017].

[5] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system”, 2008, [Online], Available: http://bitcoin.org/bitcoin.pdf, [retrieved: 09, 2017].

[6] www.blockchain.info, “Confirmed transactions per day”, [Online], Available: https://blockchain.info/charts/n-transactions [retrieved: 09, 2017].

[7] “Peercoin - Secure & Sustainable Cryptocoin.”, edited 2018, [Online], Available: https://peercoin.net/, [retrieved: 02, 2018].

[8] "PoET 1.0 Specification”, edited 2017, [Online], Available: https://sawtooth.hyperledger.org/docs/core/releases/latest/architecture/poet.html, [retrieved: 02, 2018].

[9] K. Olson, M. Bowman, J. Mitchell, S. Amundson, D. Middleton, and C. Montgomery, "Sawtooth: An Introduction", 2018, [Online], Available: https://www.hyperledger.org/wp-content/uploads/2018/01/Hyperledger_Sawtooth_WhitePaper.pdf, [retrieved: 02, 2018].

[10] D. Hopwood, S. Bowe, T. Hornby, and N. Wilcox, "Zcash Protocol Specifcation", edited 2018, [Online], Available: https://github.com/zcash/zips/blob/master/protocol/protocol.pdf, [retrieved: 02,2018].

[11] V. Buterin, “A next-generation smart contract and decentralized application platform”, 2013, [Online], Available: https://github.com/ethereum/wiki/wiki/White-Paper, [retrieved: 09, 2017].

[12] Iroha, “White Paper”, 2016, [Online], Available: https://github.com/hyperledger/iroha/blob/master/docs/iroha_whitepaper.md , [retrieved: 09, 2017].

[13] Ripple, “Solution Overview”, 2013,[Online], Available: https://ripple.com/files/ripple_solutions_guide.pdf, [retrieved: 09, 2017].

[14] “Welcome to Hyperledger Fabric”, edited 2017, [Online], Available: https://hyperledger-fabric.readthedocs.io/en/latest/, [retrieved: 09,2017].

[15] “The Monax Platform”, edited: 2017, [Online], Available: https://monax.io/platform/, [retrieved: 09, 2017].

[16] “Openchain - Blockchain Technology for the Enterprise”, edited: 2017, [Online], Available: https://www.openchain.org/, [retrieved: 09,2017].

[17] G. Greenspan, “MultiChain Private Blockchain — White Paper”, 2015, [Online], Available: https://www.multichain.com/white-paper/, [retrieved: 09, 2017].

[18] N. Szabo, “Smart Contracts: Building Blocks for Digital Markets,” 1996.

[19] G. Greenspan, “Why many smart contract use cases are simply impossible”, 2016, [Online], Available: https://www.coindesk.com/three-smart-contract-misconceptions/, [retrieved: 09, 2017].

[20] “Involver – Social Volunteering”, 2015, [Online], Available: www.getinvolver.com, [retrieved: 09, 2017].

[21] A. Marczewski, “Gamification: a simple introduction”, 2013, [Online], Available:

https://books.google.se/books?id=IOu9kPjlndYC, [retrieved: 09, 2017].

[22] K. Huotari and J. Hamari, "Defining gamification: a service marketing perspective", in Proceeding of the 16th International Academic MindTrek Conference, ser. MindTrek '12. New York, NY, USA: ACM, 2012, pp. 17–22, doi:10.1145/2393132.2393137.

[23] S. Deterding, D. Dixon, R. Khaled, and L. Nacke, "From game design elements to gamefulness: Defining gamification", in Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments, ser. MindTrek '11. New York, NY, USA: ACM, 2011, pp. 9–15, doi:10.1145/2181037.2181040.

[24] "Duolinguo", edited 2017, [Online], Available: https://www.duolingo.com/, [retrieved: 09, 2017].

[25] "Fitocracy", edited 2017, [Online], Available: https://www.fitocracy.com/, [retrieved: 09, 2017].

[26] J. Hamari, J. Koivisto, and H. Sarsa, "Does gamification work? – a literature review of empirical studies on gamification", in the 47th Hawaii International Conference on System Sciences, Jan 2014, pp. 3025–3034.

[27] M. H. Phillips and L. C. Phillips, "Volunteer motivation and reward preference: an empirical study of volunteerism in a large, not-for-profit organization", SAM Advanced Management Journal, 2010, pp. 12–19.

[28] Wikipedia, the free encyclopedia, "Wikipedia:Barnstars", 2004, [Online], Available: https://en.wikipedia.org/wiki/Wikipedia:Barnstars, [retrieved: 09, 2017].

[29] "Foursquare", edited 2017, [Online], Available: https://www.foursquare.com, [retrieved: 09, 2017].

[30] "X Box Live", edited: 2017, [Online], Available: https://www.xbox.com/en-US/live, [retrieved: 09, 2017].

[31] A. Anderson, D. Huttenlocher, J. Kleinberg, and J. Leskovec, "Steering user behavior with badges", in Proceedings of the 22Nd International Conference on World Wide Web, ser. WWW '13. New York, NY, USA: ACM, 2013, pp. 95–106, doi:10.1145/2488388.2488398.

[32] M. Montola, T. Nummenmaa, A. Lucero, M. Boberg, and H. Korhonen, "Applying game achievement systems to enhance user experience in a photo sharing service," in Proceedings of the 13th International MindTrek Conference: Everyday Life in the Ubiquitous Era, ser. MindTrek '09. New York, NY, USA: ACM, 2009, pp. 94–97 doi :10.1145/1621841.1621859.

[33] J. Hamari and V. Eranti, "Framework for Designing and Evaluating Game Achievements," in Proceedings of DiGRA 2011 Conference: Think Design Play, 2011, [Online], Available: http://www.digra.org/wp-content/uploads/digital-library/11307.59151.pdf, [retrieved: 09, 2017].

[34] "The solidity programming language", 2015, [Online], Available: https://github.com/ethereum/wiki/wiki/The-Solidity-Programming-Language, [retrieved: 09, 2017].