An Industrial Prototype of Trusted Energy Performance Contracts using Blockchain Technologies

Önder Gürcan[†] and Marc Agenis-Nevers[‡] and Yves-Marie Batany[‡]
and Mohamed Elmtiri[‡] and François Le Fevre[†] and Sara Tucci-Piergiovanni[†]

***Abstract*— Energy conservation measurements in buildings are more and more popular as they benefit from an intelligent contractual framework called Energy Performance Contracts (EPC), where energy savings are measured as the difference between a predictive baseline model and the actual consumption. While modern predictive models make use of large amounts of data from external sources and increasingly complex algorithms, these two aspects make their use difficult in practice because they need mutual understanding and transparency, requiring the involvement of a third-party for auditing. In this sense, we designed and developed a prototype that overcomes these issues by storing the predictive models and the data in an immutable blockchained data structure using the Ethereum framework. To the best of our knowledge, this is the first working prototype using the blockchain technology applied to EPCs. This paper presents and discusses the technical solutions and best-practice guidelines adopted in this prototype.**

I. Introduction

In an always more constrained environment (rising energy cost, intensification of the environmental regulations, reduction of the margins) building owners are seeking to reduce building energy consumption and the related costs. To meet the customer needs, Energy Services Companies (ESCO) provide to *clients* Energy Performance Contracts (EPC), which can be defined as contracts under which energy savings are provided, verified and monitored during the whole term of the contract[1]. More in detail, the ESCO, also called *service provider*, designs and implements so-called Energy Conservation Measures (ECMs) (for instance window replacement with modern insulation, lighting, etc.) and guarantees a given level of energy savings over the period of the agreement.

Energy savings secure the financial revenue that is used to fund the cost of improvements and services incurred at the service provider side. Note that financial savings are in general shared between the two parties from the start. Once the costs have been repaid, the client should be able to keep the full savings generated from the ECMs. The duration of the savings guarantee will typically last until the project costs have been covered. In case of a failure in provisioning the contractually-agreed energy savings, financial penalties are applied to the service provider which reduces the contract revenue. For customers, EPC has become an approach to retrofit sites (buildings or industrial) with energy savings and energy generation improvements, and to reduce carbon emissions.

Since energy savings cannot be directly measured, a so-called measurement and verification (M&V) procedure[2] is at the cornerstone of an EPC. The M&V procedure is the process of using measurement to determine reliably the actual savings created within a facility. Concretely, the energy saving is computed as the difference between a predictive baseline model and post-installation real energy consumption over the period of use. A poor basis for M&V procedure can create problems such as an unfair allocation of performance risk as well as savings calculations being unclear or taken for an inappropriate baseline. M&V procedure can be complex since different factors, such as weather or building occupancy, need to be taken into account during the lifetime of the contract.

The International Performance Measurement and Verification Protocol (IPMVP) has been developed over many years to provide guidance and expected standards for an M&V procedure[3]. IPMVP covers the way variables (explanatory or static) are included in the baseline predictive model and how adjustments must be made (e.g., if building occupancy rises). This protocol is used by many EPC projects as the basis for M&V procedure and it is important that clients understand the proposed M&V approach before committing to such contracts. In particular, the need for understandability when it comes to baseline modeling and data sourcing often leads the client to favor the use of simple predictive models. However, more complex models using a wider set of data increase prediction quality and hence decrease the commitment risk.

We illustrate some problems that can arise in the operational context:

- the weather provider might retroactively modify the data used for the baseline,
- the client might modify the building set-points or equipment without notifying the ESCO,
- the prediction values from a complex model might vary depending on the version used.

[†]Ö. Gürcan, F. Le Fevre and S. Tucci-Piergiovanni are with CEA, LIST, the Laboratory for Trustworthy, Smart, Self-Organizing Information Systems, 91191 Palaiseau, France name.surname at cea.fr

[‡]M. Agenis-Nevers, Y.-M. Batany and M. Elmtiri are with Veolia Research and Innovation (VERI), Centre de Recherche de Limay, 78520 Limay, France name.surname at veolia.com

[1]M&V Guidelines: Measurement and Verification for Performance-Based Contracts Version 4.0, November 2015, https://www.energy.gov/sites/prod/files/2016/01/f28/mv_guide_4_0.pdf, last access on 25/05/2018.

[2]International Performance Measurement and Verification Protocol (IPMVP), 1997, https://www.nrel.gov/docs/fy02osti/31505.pdf, last access on 25/05/2018.

[3]International Performance Measurement and Verification Protocol (IPMVP) Vol I. Efficiency Valuation Organization, 2007.

Energy agencies have developed software to allow the computation of M&V models in a more controlled environment [1] but they do not solve the other trust issues. In this context, the present work evaluates the relevance of blockchain technologies to overcome the aforementioned limitations and to gain client trust even in presence of sophisticated, and then not immediately understandable, M&V procedures. If effective, blockchain-enabled EPC can lead to massive adoption of EPCs and widespread energy savings.

Blockchain technologies indeed provide an incorruptible digital ledger that can be programmed to record any kind of data and calculations (e.g., measurement of real effects, operating conditions, reference situation) in a multi-stakeholder environment (e.g., operator, project owner, users). More specifically, blockchains guarantee that the recorded data is immutable, thanks to cryptographic techniques, and highly replicated thanks to distributed consensus protocols [2], storage or trusted third parties.

This technology could be useful for operating EPC without the need of a trusted third party such as a government agency or independent consultants. A prototype dedicated to this application has been developed to evaluate whether the blockchain technology is able to provide the required level of trust between the client and Veolia as part of energy performance contracts commitment during the following M&V phases:

- qualification of the data,
- automatic calculations of the savings,
- verification of commitments.

The use of the blockchain occurs once a baseline model has been established and fit on historical data. We designed the blockchain so that the prediction step and all related calculations would happen inside it. Since the predictive model was developed with the R software[4], it is important to note that it is not possible to execute the R script directly in the blockchain, and we developed the equivalent code of the R script based on the blockchain smart contract layer.

A. Contributions

This article is an experience report presenting the design and development of a blockchain-enabled EPC prototype. The development methodology followed a classical approach, even though features and limitations specific to the target blockchain technology influenced the design of the solution. Interestingly, due to performance issues, the design has to be refined in a set of very small software modules, called smart contracts in the blockchain jargon. Moreover, we faced immediately the issue of having a flow of data coming from outside the chain. While the blockchain guarantees data immutability, data veracity is not guaranteed, i.e. a false data coming from outside can be in principle stored in the blockchain; redundancy and voting mechanisms have been put in place to solve this issue. Another important, more technological aspect, is about the non-availability of very basic libraries to treat calculation of predictive models. Finally, we spot the need of having an innovative approach for smart contracts management, that is not only able to translate the contractual terms and agreements into smart contract codes, but also able to flexibly accommodate amendments to the terms of use.

[4]R is an open source project of statistical language. `https://cran.r-project.org/`. Last access on 23/03/2018.

B. Organization

Section II presents in more details energy performance contracts. Section III specifies how smart contracts can be used for a blockchained EPC. Section IV describes the developed prototype application from analysis to test. Section V discusses and concludes the paper.

II. Energy Performance Contracts

Energy performance contracts (EPCs) aim at improving the energy efficiency of a set of facilities. The energy consumption of a facility is denoted as $E[t]$ where t is the discrete time. An EPC involves the client and the an energy service company (contractor). Before the contractualization at time t_0, a predictive model (or baseline) of energy consumption is constructed based on a historical set of data. The predictive model is denoted $\mathcal{F}$ and it explicitly gives a relation between the energy consumption $E[t]$ and a set of explanatory variables $X[t] \in \mathbb{R}^N$ such that

$$E_b[t] = \mathcal{F}(X[t]). \tag{1}$$

Temperature or humidity are often used as explanatory variables. During contractualization, the energy savings are computed for a period T as

$$S[t] = \sum_{t-T}^{t} E_b[t] - E[t] \tag{2}$$

and are shared between both parties. Figure 1 shows the timeline of an EPC with our notations. After contractualization, the savings belong to the client. The savings are generally computed on a basis of *one month* and financial transactions occurs once a year. As a result, predictive models are generally *monthly* computed. It is easy to see why the choice of the predictive model is of crucial matter: the savings are computed from this estimation and errors on the predictive model will be directly reverberated on the savings.

The way to model energy savings while following the M&V methodology has been described for more than twenty years [3], [4]. The literature related to statistical models for buildings consumption prediction suggests a very wide range of models $\mathcal{F}$ with no limitation: generalized linear models [5], machine learning, time-dependent models [6], Bayesian models [7], Gaussian processes [8], etc. However, in the restrictive contractual context of EPCs, only a subset of these statistical techniques are used - almost exclusively linear models with all their variants: polynomial and segmented models, multivariate and even robust regressions; the time stamps encountered are mostly monthly, hourly and daily [9].

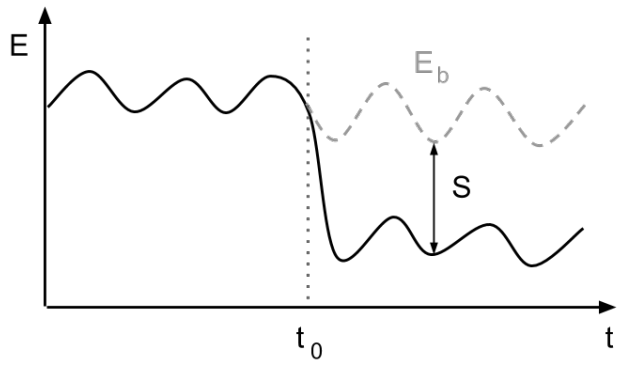

Fig. 1. The time line of an energy performance contract. The moment of contractualization is denoted t_0.

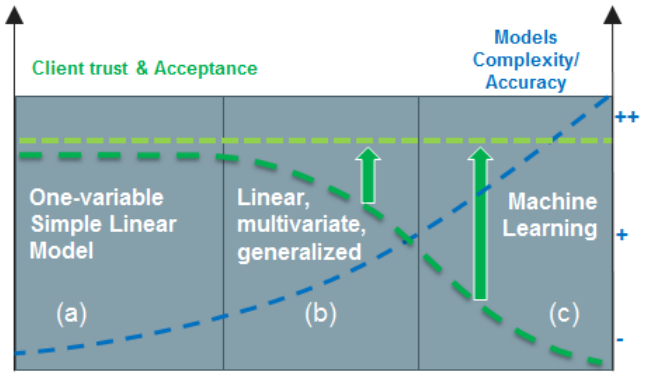

Fig. 2. Model complexity versus client acceptance diagram. The green vertical arrows symbolize the expected impact of the blockchain. Note: the blue dashed curve assumes that the complexity does not yet reach a point where the model over-fits the data. If so, the accuracy would go down.

While implementing such models, the problem of gaining the client's trust arises. Though the IPMVP framework mentions no restriction on the model type, and EVO (Efficiency Valuation Organization) experts suggest that there is no reason *a priori* to put non-linear forms aside, the need for understandability or interoperability becomes merely a matter of trust: the question arise whether the client will accept a black-box model that cannot be reproduced on an Excel sheet or not [10]. Figure 2 shows empirically how client trust and acceptance evolves with model complexity. In the field, most EPCs still rely today on 1-variable OLS linear models (Figure 2a), leaving a huge room for complexity improvement (Figure 2b or c) before over-fitting. Figure 2 shows empirically how trust evolves with model complexity.

III. Blockchains and Smart Contracts

To tackle the problems given in subsection II, we introduce Trusted Energy Performance Contracts (TEPCs), i.e. storing and managing the predictive models and the data used using the blockchain technology, since it guarantees the immutability of the information once written on it. The blockchain will substitute for a trusted third-party, a role that is usually played by expert consultants who help the client validate contractual models.

The blockchain technology is first proposed by the Bitcoin protocol [2]. A blockchain system is a network of nodes in which each node keeps a replication of an immutable append-only ledger of transactions [11]. The transactions are issued by nodes called *users* to exchange information, e.g., to exchange bitcoins in the Bitcoin protocol. The copy of all issued valid transactions are stored locally in the same order in a local ledger by all nodes (replication). The reliability of the replication process is ensured by special nodes called *miners* that collectively build the ledger as a chain of blocks that are interconnected by cryptographic links. This ensures that once the information has been appended, it will no longer be modified by anyone in the future.

Smart contracts are contracts (or any kind of program) that are converted to deterministic computer code and, stored and replicated on blockchain systems [12]. They are executed exactly in the way by each node in the blockchain system. In other words, the function call requests to them are ordered (thank to the consensus mechanism) and then executed in the same order, sequentially, one request at a time, at all nodes (this is also called *active replication* [13]).

Developing TEPCs as smart contracts is not trivial and requires a new framework including four major phases:

1) *Definition of TEPC.* In this phase, all stakeholders agree on the baseline (predictive model), usage conditions (e.g., frequency of feeding data), the price of the energy and the data qualification process.
2) *Programming the corresponding smart contracts.* The conditions agreed on the first phase are hard-coded inside the smart contracts, which makes them unchangeable from outside.
3) *Deployment into the blockchain system.* In this phase, a unique address for each stakeholder is generated and hard-coded inside the smart contracts, to allow controlled access to the smart contracts. Then the smart contracts are deployed to the blockchain and their corresponding unique addresses are generated.
4) *Execution inside the blockchain.* In this phase, the smart contracts are executed only if they are called through their addresses.

If at any time, the stakeholders want modifications of the smart contracts, a new smart contractualization process should be started from scratch.

IV. The Prototype for Trusted EPCs

The aim of this prototype is to develop a trusted energy performance contract (TEPC) that use a multivariate linear predictive model (Figure 2b) for an entire building located in a part of the world where consumption mainly consists of air cooling loads.

The architecturally significant requirements of this prototype are captured using the FURPS+ system for classifying requirements [14]: the functional requirements are identified in Section IV-A, the reliability requirements are identified in Section IV-B and the implementation requirements are identified in Section IV-C. Based on these requirements, an architectural design is then made in Section IV-D. Lastly,

Section IV-E presents the implementation details of the prototype and Section IV-F presents how the prototype is tested using some real data and the results obtained.

A. Functional Requirements

Functional requirements represent what the main product shall do. To determine the functional requirements of the prototype, two main uses cases are identified.

1) The Daily Prediction Scenario: In this scenario, the authorized sources *Client*, *Veolia* and *Meteo France* provide their climate information (*temperature*, *humidity* and *pressure*) to the prototype every hour. At the end of the day (exactly at 23:59), the authorized source *electricity meter of Client* provides a single *daily energy consumption* value and the authorized sources *sensors in the Client building* provide *occupancy* and *air conditioning temp*. Immediately after, the prototype first makes a daily qualification process: for each hour, a qualified hourly sample is computed using the data at hand and is then added to the list of qualified samples. When the whole calculation for that day is finished, the prototype calculates a daily saving value by applying a *bivariate linear model* based on temperature and humidity, where temperature is processed to represent a *cooling degree day* (sum of hourly temperature exceeding 27°C) [15]. The prediction model, fitted with *R*, gets its coefficients hard-coded into simple mathematical equations. The saving value is then accumulated to the monthly saving. At the end of the month, the prototype computes the monthly savings as the difference between predicted and actual consumption.

2) The Audit Scenario: In this scenario, an authorized user (i.e. *Client*, *Veolia* or *Meteo France*) consults the past information (the log stored in the blockchain) for auditing purposes. Different types of auditing requests are possible, such as *audit the prediction of the consumption for 12:00 on 21/03/2018*. The results include the climatic data provided by the sources for the prediction of daily saving, the qualified climate values together with their degrees of reliability and the authorized sources that provided them (see Figure 4).

B. Reliability Requirements

Blockchains guarantee that "once the data has been registered, it will no longer be modified by anyone", (immutability). However, they do not guarantee the "veracity" of the registered information. As a result, dedicated data qualification processes shall be defined. They shall then be contracted with the client and hard-coded as smart contracts.

One qualification process, which is called *filtering*, shall check the boundary values for any given data. For example, temperature values shall be between -30°C and 60°C, humidity values shall be between 0% and 100%, pressure values shall be between 850 bar and 1060 bar and consumption values shall be higher than or equal to 0 kW. If, for any reason, a value out of these boundaries is provided, it shall be changed to the closest boundary value. For instance, if a temperature value of -40°C is provided, it should be changed as -30°C and stored as such.

For the data that are provided by multiple sources, the qualification process shall use a *voting* mechanism that is based on the level of confidence of each authorized source to decide the qualified value and its reliability level.

C. Implementation Requirements

Implementation requirements specify or constrain the coding or construction of a system. An implementation the prototype requires the following: an underlying blockchain technology that has built-in smart contract support and fast-prototyping feature. To this end, the Ethereum blockchain platform[5] is chosen. In the following, Ethereum is first described, then the limitations of this choice are discussed.

1) The Ethereum Blockchain Platform: Ethereum is a blockchain platform that runs smart contracts. Smart contracts run on every computer in the Ethereum network and they can read/write data, do expensive computations like using cryptographic primitives, make calls (send messages) to other contracts, etc. Consequently, there is mechanism called *gas* to limit the resources used by each contract. So-called *gas* is used for measuring the cost of the smart contract operations, and each gas unit consumed by a transaction (function call) must be paid for in Ether, based on a gas/Ether price which changes dynamically. This price is deducted from the Ethereum *account* sending the transaction. Transactions also have a gas limit parameter that is an upper bound on how much gas the transaction can consume, and is used as a safe-guard against programming errors that could deplete an account's funds.

Smart contracts in Ethereum are written in the Solidity language[6]. Solidity smart contracts are similar to classes in other programming languages: they have states and functions. There are two types of functions that can appear in a smart contract:

- *Read-only functions*: functions that do not do any state changes. They only read state, perform computations, and return values. As these functions can be resolved locally by each node, they cost no gas.
- *Transactional functions*: functions that do a state change in the contract or move funds. As these changes need to be reflected in the blockchain, transactional function execution requires sending a transaction to the network and spending gas.

After smarts contracts are developed, the next issue is to choose the right Ethereum network for deployment. From the deployment point of view, there are three types of networks available for Ethereum applications: the public network (i.e. Main Network), public test networks (e.g., Morden, Rinkeby) and private networks. The Main Network is where the production applications are deployed and is accessible by anybody. Test networks are where the test applications that

[5]Ethereum Foundation. Ethereum's white paper. `https://github.com/ethereum/wiki/wiki/White-Paper`, 2014. Last access on 20/03/2018.

[6]Ethereum Foundation. The solidity contract-oriented programming language. `https://github.com/ethereum/solidity`. Last access on 20/03/2018.

do not have confidential data/algorithms are deployed and is accessible by anybody. Private networks enable to create one or more Ethereum nodes that are only accessible to people with permissions. Private networks are convenient for testing purposes or cases involving highly sensitive data, which is the case of our prototype (confidentiality of the daily prediction algorithm and all hourly samples). A private network was therefore chosen for the present application.

2) Technical Difficulties: There are three main types of technical obstacles when implementing a R-based predictive model as a Solidity smart contract.

- *Lack of floating-point number support*: Solidity does not support floating point numbers[7]. For example, the operation $1.9 * 2$ truncates the result to 2 and the operation $-1.9/2$ truncates the result to 0. To tackle this limitation, the units of the values can be changed to small enough unit (e.g., from kW to watts) to produce the results of desired precision.
- *Lack of standard math libraries*: Solidity lacks the standard mathematical libraries needed for implementing predictive R models as smart contracts, especially the ones about array and matrix operations. Thus, necessary operations shall be implemented as Solidity functions.
- *Memory limitations*: Solidity smart contracts have certain memory limitations. As a result, the original *monthly predictive R model* shall be transformed into a *daily predictive model* since it is not possible to keep one month hourly climate data inside a smart contract.

Another important limitation of Ethereum, and all other existing blockchain technologies, is the accuracy of the time. In blockchain systems, it is only possible to access to the time information by using the time-stamps of blocks. However, this information is not totally reliable and can be easily manipulated over short periods by an attacker. Moreover, smart contract functions can only be activated from an external call, i.e. timers cannot be implemented in smart contracts for triggering specific functions at specific times. Therefore the time-stamp information of blocks needs to be supplemented with some other strategy in the case of high-value/risk applications, such as in our prototype where to issue invoices to the clients the calculations should be made at precise times, i.e. the end of the day and the end of the month.

Based on the requirements identified in Section IV-A, Section IV-B and Section IV-C, the next section describes the architectural design of the prototype.

D. Design

This section describes the architectural design of the prototype as a functional architecture (Figure 3). A functional architecture is an architectural model from a usage perspective which is composed of the modules (blocks) that represent each software entities and flows that represent the interactions between these entities.

The functional system architecture of the prototype is composed of the following modules (Figure 3): the blockchain (the red block), the smart contracts (the blue blocks), the authorized sources (yellow blocks), the auditor (the grey block) and the user interface (the green block).

1) The Blockchain: The underlying blockchain network is a private Ethereum blockchain network. It is composed of the following: an immutable ledger `Blockchain`, the smarts contracts `Data Qualifier`, `Predictor`, `Aggregator` and `Validator`, and 7 predefined wallets (`account0`, `account1`, ... `account6`), one for the miner and six for the authorized sources, with sufficient balances. When the Ethereum network is initialized, first a miner that is using `account0` (i.e. coinbase) is activated. This makes the miner create blocks for `Blockchain`. After that, the smart contracts are deployed to `Blockchain`. Since the dedicated wallets (account addresses) for each authorized source are hard-coded in these smart contracts, once the smart contracts are deployed, it is not possible change the information about the authorized sources in the blockchain. This approach increases the confidence to the prototype.

2) Authorized Sources: The authorized sources are the sources that are authorized to provide data to the system. Each authorized source can only provide the data expected from them. There are six authorized sources in the system:

- `Client Meter` is an electricity meter that provides the actual daily consumption of the client (in terms of kW).
- `Client` module provides hourly samples gathered by the sensors installed in the client's building.
- `Meteo France` module provides hourly samples gathered by the agreed third-party like Meteo France.
- `Veolia` is a software entity that provides hourly samples gathered by the external contractor Veolia.
- `Time Oracle` is a software entity that provides precise time information to the blockchain such as the end of the day or the end of the month. It is added to the design to tackle with the correct time information problem described in Section IV-C.2.
- `Client Building` represents the sensors installed in the client's building and provides occupancy and air conditioning temperature data (static data).

3) Smart Contracts: Due to the limitations mentioned in Section IV-C, a *trusted EPC* is represented as four smart contracts in the system: `Data Qualifier`, `Predictor`, `Aggregator` and `Validator`.

- `Data Qualifier`: The aim of this smart contract is threefold: (1) to store information about authorized sources, (2) to qualify the data coming from authorized sources as described in Section IV-B and (3) to start the daily prediction process. (1) is fulfilled by hard-coding the authorized sources inside the contract code and (2) is fulfilled by providing dedicated transactional functions for *filtering* and *voting*. The voting algorithm works as follows: `Client`, `Veolia` and `Meteo France`

[7]Neither Solidity nor other smart contract languages support floating-point numbers. This is because the result of floating-point calculations depend on the used algorithms and/or the CPU architectures [16], and such a case is not supported in active replication [13].

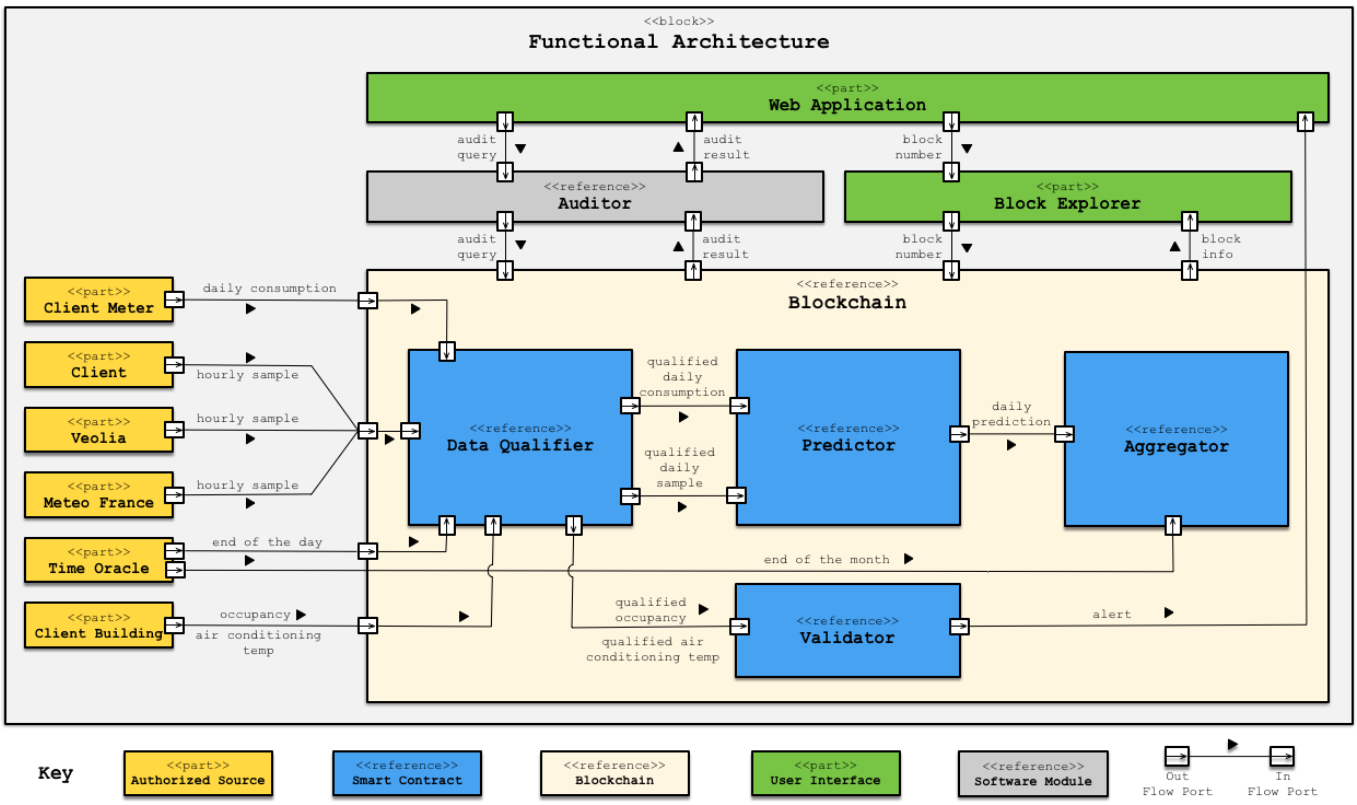

Fig. 3. The functional system architecture represented as a SysML block diagram. All data coming to contracts are logged in `Blockchain`.

have predefined *weights* (3, 3, and 4 respectively) that represent their degrees of reliability. The voting algorithm first groups the closest inputs coming from two authorized sources together. Then it selects the group that has the highest weight. Lastly, it calculates the weighted average of the selected group as the qualified value and calculates the sum of the weights as the reliability value. Note that the choice of weights distribution makes the neutral source prevail unless it is contradicted by the two others. (3), on the other hand, is triggered by the time_oracle at the end of the day (23:59) and fulfilled by passing all the qualified hourly values, i.e. qualified daily sample, to `Predictor` for prediction calculation.

- `Predictor`: The aim of this smart contract is to predict the daily saving using the qualified daily sample for a given date. When a qualified daily sample is handled to `Predictor`, the daily saving is calculated by using the statistical model separately fitted with R. The output of this daily saving calculation is then passed to the `Aggregator` smart contract.
- `Aggregator`: The aim of this smart contract is to aggregate savings from daily into monthly time stamp. When a daily saving is added, it increments the total saving by the amount of the current date. Eventually, when the `Time Oracle` declares the end of the month, the contract calculates the monthly saving in kW and translates it into a financial flow.
- `Validator`: The aim of this smart contract is to validate static variables of the client building.

4) Auditor: To query `Blockchain` for audition purposes as described in Section IV-A.2, a software module called `Auditor` is used. The authorized users make their audition queries through this module and then the retrieved information is shown to the corresponding authorized user through a user interface.

5) User Interface: To provide a human machine interface between the authorized users and the prototype, two user interface modules are used: `Web Application` and `Block Explorer`. `Web Application` user `Auditor` to query `Blockchain` and display the corresponding results. Since the audited information is stored in `Blockchain`, it shows all results together with their corresponding block numbers. The authorized users may then check the details of each information by using these block numbers through `Block Explorer`.

E. Implementation

The implementation is done using the following technologies: the blockchain by using Ethereum[8], the smart contracts by using Solidity, the authorized sources and the auditor by

[8]Ethereum is the first and leading blockchain framework that allows users to develop, run and use smart contracts.

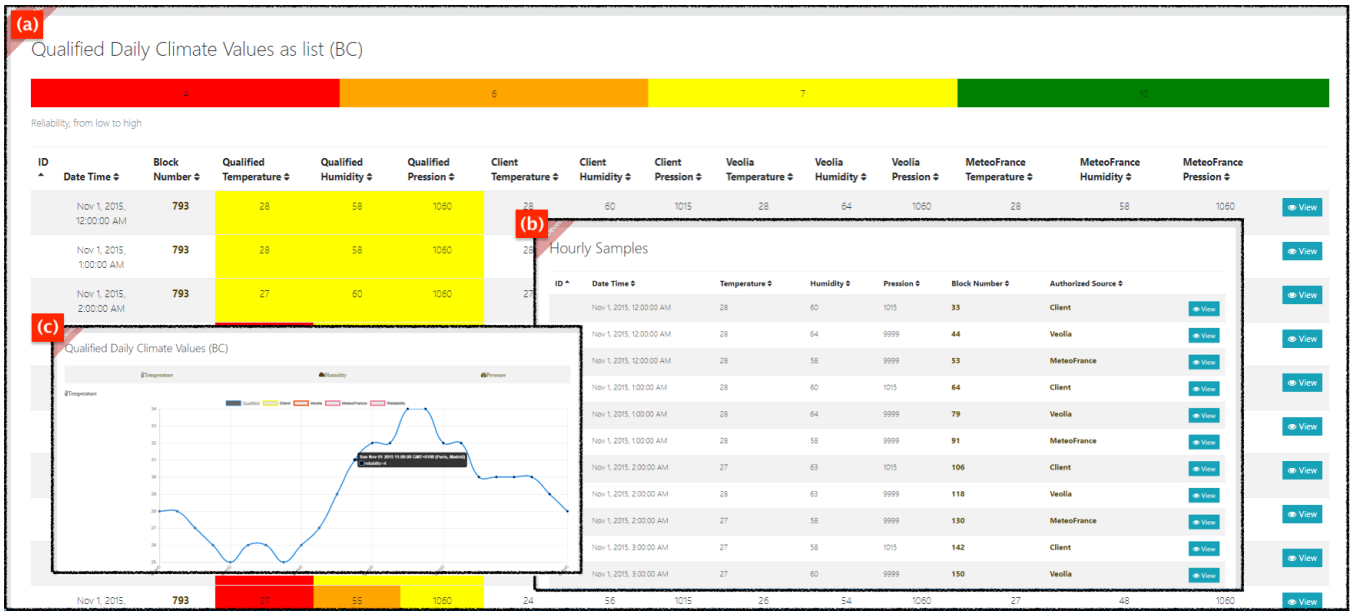

Fig. 4. The screen-shots that belong to three different windows of `Web Application`: (a) Qualified Daily Climate Values as List, (b) Hourly Samples, and (c) Qualified Daily Climate Values as diagram.

using web3j[9], and the UI by JHipster[10] and REST[11].

Since there is no standard library for mathematical operations, especially for array operations, in Solidity, all necessary mathematical function have been implemented from scratch. Concretely the following functions are implemented:

- `pmax(int p, int[] v)`: returns the `maxOf(p, v[i])` for each element `i` of a given array `v`.
- `expArray(int[] v)`: returns the exponential of each value in a given array `v`.
- `expScalar(int s)`: returns the result of the exponential function of the form e^s.
- `power(int a, int b)`: returns the result of the power function of the form a^b.
- `factorial()`: returns the factorial of a given scalar.
- `xScalar(int[] v, int s)` functions: used for adding to, subtracting from, multiplying with or dividing by a given vector `v` and given scalar value `s`.
- `xArray(int[] v, int[] y)`: returns the addition, subtraction, multiplication or division of two given arrays with the same size.
- `sum(int[] v)`: returns the sum of the all elements in an int array.
- `mean(int[] v)`: returns the mean of the values of a given integer array.
- `max(int[] v)`: returns the greatest value of a given integer array.
- `maxOf(int a, int b)`: returns the max value of two given scalar values.

To tackle with the floating-point number support limitation, the original predictive *R* model is be transformed into an integer-based model. To do so, a multiplicative factor is used and dedicated functions are developed in *R* to simulate arithmetic operators that discard the fractional part of both the input and the output. The deviation from the true monthly savings value is +1.3% or -0.0006% with a multiplicative factor of 10^3 or 10^6 respectively.

F. Test

The developed prototype is tested by using a full set of 5 days data (from 01/11/2015 to 05/11/2015), i.e. climate values provided by each authorized source. These values are partly real (energy consumption and meteo_france weather data) and partly simulated (the two other weather sources were artificially altered to show a situation of divergence and test the reliability procedure).

Figure 4 shows the results of the conducted test[12]. Figure 4a and Figure 4c show the same qualified climate values stored in `Blockchain` with different views. Each qualified climate value has an associated color that represents its degree of reliability: the red color means the reliability is 4, the orange color means the reliability is 6, the yellow color means the reliability is 7 and the green color means the reliability is 10, when all actors agree on the climate values.

Figure 4b shows the climate values for specific dates and times (i.e., hourly sample) provided by each authorized

[9]web3j is a highly modular, reactive, type safe Java and Android library for working with Smart Contracts and integrating with clients (nodes) on the Ethereum network. This allows working with the Ethereum blockchain, without an extra overhead of having to write an integration code for the platform. `https://web3j.io/`. Last access on 21/03/2018.

[10]JHipster is a fully Open Source, widely used application generator for easily creating high-quality Spring Boot + Angular projects. `https://www.jhipster.tech/`. Last access on 21/03/2018.

[11]REST (Representational State Transfer) is an architectural style for designing distributed systems. `https://spring.io/understanding/REST`. Last access on 21/03/2018.

[12]Due to the space limitations, the windows are shown in one single figure.

source together with their block numbers and is used to further analyze the qualified data shown in Figure 4a and 4c. For example, the details of the first line of (a) can be seen as the three first lines of Figure 4b. From any windows shown in Figure 4, upon clicking on block numbers or authorized source names, it is possible to open `Block Explorer` to manually verify the blockchained data.

V. Discussions and Conclusions

In this study, an energy performance contract (EPC) that use a multivariate linear predictive model empowered by blockchain technologies has been developed with the aim of improving the *trust* between the client and the service provider. Ultimately, such a system could allow the use of more complex predictive models. To the best of our knowledge, this is the first working EPC prototype using blockchain technology performing the following value-added services:

- the invoicing service based on (1) the data flow in the monitoring phase and the associated qualification method and (2) the baseline model used for prediction.
- the audit service that supports querying the blockchain about the invoicing process.

This working prototype shows the feasibility of the approach, however, several limitations must be still addressed. Memory and decimal limitations make it hard, for now, to implement a machine learning model as a smart contract, even if the fitting part is kept outside the Blockchain. Some machine learning models render a couple of simple prediction rules or equations [10] (such as decision trees, regularized regression, generalized linear regression) and can be regarded as "Ethereum-compatible", but the majority depend on large matrices or heavy computations [17]: SVM, ensemble methods (boosting, randomForest), neural networks, KNN. The issue of how to implement a numerically complex predictive model, making them more "trustworthy" to a client (Figure 2c), has not yet been addressed.

As future work, we will explore the following axes:

- Evaluating alternative blockchain technologies, as Hyperledger Fabric[13] and Monad/Tendermint[14] that enjoy a different execution mechanism for smart contracts.
- Possibility to treat the code as a "data" to be stored in the blockchain and execute calculations outside the blockchain. Inputs and outputs (and some potential intermediate checkpoints) are stored in the blockchain. It would be possible with this mechanism to reconstruct at any time the execution trace of the off-chain calculation.

Another important subject is the life-cycle management of smart contracts when the real contracts between the client and the service provider are supposed to evolve. Up-to-now smart contracts are intended to freeze in a piece of code conditions and terms, which are then secured and nobody can change (since the contract code is secured by the blockchain once deployed). When an amendment to a given contract is made, terms and conditions get obsolete and a new smart contract should replace the old one, that is the old one should be deactivated. Methods for deactivation while guaranteeing security (no malicious manipulation of terms and contracts and no access to obsolete versions) will be studied.

The last point is about the concrete deployment that should be pursued. The prototype is today running in a private small-scale network for demonstration purposes only. A real experimentation must be conducted as next step and the choice of a private or public network must be done before commercialization. This choice is indeed also related to confidentiality issues, revenues models and the possibility of creating a community of users, i.e. both clients and service providers in the energy domain. As a conclusion, even if there is still a long way to go before actual commercialization, our experience reported in this paper highlights some exciting R&D issues that must to be addressed in the next future to make blockchain technology industrial-ready.

[13] `https://www.hyperledger.org/projects/fabric`. Last access on 23/03/2018.

[14] `https://tendermint.com/`. Last access on 23/03/2018.

References

[1] D. J. Taasevigen, S. Katipamula, and W. Koran, "Interval data analysis with the energy charting and metrics tool (ecam)," tech. rep., Pacific Northwest National Laboratory (PNNL), Richland, WA (US), 2011.

[2] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2018. `https://bitcoin.org/bitcoin.pdf`.

[3] T. A. Reddy, N. F. Saman, D. E. Claridge, J. S. Haberl, W. D. Turner, and A. T. Chalifoux, "Baselining methodology for facility-level monthly energy use-part 1: Theoretical aspects," in *ASHRAE transactions*, ASHRAE, 1997.

[4] O. Akinsooto, D. De Canha, and J. Pretorius, "Energy savings reporting and uncertainty in measurement & verification," in *Power Engineering Conference (AUPEC), 2014 Australasian Universities*, pp. 1–5, IEEE, 2014.

[5] P. N. Price, "Methods for analyzing electric load shape and its variability," tech. rep., LBNL/CEC, 2010.

[6] D. Ruch, J. Kissock, and T. Reddy, "Prediction uncertainty of linear building energy use models with autocorrelated residuals," *Journal of solar energy engineering*, vol. 121, no. 1, pp. 63–68, 1999.

[7] B. Yan, *A Bayesian approach for predicting building cooling and heating consumption and applications in fault detection*. University of Pennsylvania, 2013.

[8] M. C. Burkhart, Y. Heo, and V. M. Zavala, "Measurement and verification of building systems under uncertain data: A gaussian process modeling approach," *Energy and Buildings*, vol. 75, pp. 189–198, 2014.

[9] T. Reddy, J. Kissock, S. Katipamula, D. Ruch, and D. Claridge, "An overview of measured energy retrofit savings methodologies developed in the texas loanstar program," 1994.

[10] C. Molnar, "Interpretable machine learning. a guide for making black box models explainable.." `https://christophm.github.io/interpretable-ml-book/`. Accessed: 2018-03-22.

[11] Ö. Gürcan, A. Del Pozzo, and S. Tucci-Piergiovanni, "On the bitcoin limitations to deliver fairness to users," in *On the Move to Meaningful Internet Systems. OTM 2017 Conferences* (H. Panetto, C. Debruyne, W. Gaaloul, M. Papazoglou, A. Paschke, C. A. Ardagna, and R. Meersman, eds.), (Cham), pp. 589–606, Springer Int. Publishing, 2017.

[12] N. Szabo, "Smart contracts: Formalizing and securing relationships on public networks," *First Monday*, vol. 2, September 1997.

[13] B. Charron-Bost, F. Pedone, and A. Schiper, eds., *Replication: Theory and Practice*. Berlin, Heidelberg: Springer-Verlag, 2010.

[14] R. B. Grady, *Practical Software Metrics for Project Management and Process Improvement*. NJ, USA: Prentice-Hall, Inc., 1992.

[15] R. Bonhomme, "Bases and limits to using degree. dayunits," *European journal of agronomy*, vol. 13, no. 1, pp. 1–10, 2000.

[16] D. Monniaux, "The pitfalls of verifying floating-point computations," *ACM Trans. Program. Lang. Syst.*, vol. 30, pp. 12:1–12:41, May 2008.

[17] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*, vol. 1. Springer series in statistics New York, 2001.