UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

MÁSTER UNIVERSITARIO EN
INGENIERÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE MASTER

Design and Development of a Social Choice Model Simulation for occupant welfare in Smart Buildings based on a Blockchain solution

Eduardo Merino Machuca

2019

TRABAJO DE FIN DE MASTER

Título:	Diseño y Desarrollo de un Modelo de Simulación de Elección Social para el bienestar de los ocupantes en Edificios Inteligentes basado en una solución de Blockchain
Título (inglés):	Design and Development of a Social Choice Model Simulation for occupant welfare in Smart Buildings based on a Blockchain solution
Autor:	Eduardo Merino Machuca
Tutor:	Carlos Ángel Iglesias Fernández
Ponente:	-
Departamento:	Departamento de Ingeniería de Sistemas Telemáticos

MIEMBROS DEL TRIBUNAL CALIFICADOR

Presidente:	—
Vocal:	—
Secretario:	—
Suplente:	—

FECHA DE LECTURA:

CALIFICACIÓN:

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN

Departamento de Ingeniería de Sistemas Telemáticos
Grupo de Sistemas Inteligentes

TRABAJO DE FIN DE MASTER

Design and Development of a Social Choice Model Simulation for occupant welfare in Smart Buildings based on a Blockchain solution

Junio 2019

Resumen

Las ciudades y los edificios inteligentes han despertado recientemente una gran atención. Estas tecnologías proporcionan entornos inteligentes considerando el uso de tecnologías como la Internet de las Cosas y la Inteligencia Ambiental. Los entornos inteligentes permiten mejorar la calidad de vida y el bienestar de las personas, satisfaciendo sus demandas y necesidades.

Para ello, los entornos inteligentes pueden personalizar y mejorar sus servicios teniendo en cuenta las opiniones, elecciones y preferencias de los usuarios. Este proceso se conoce como el empoderamiento de los usuarios o 'democratización'. Además, los entornos inteligentes proporcionan normalmente servicios compartidos. Por tanto, deben ser capaces de maximizar el bienestar de múltiples usuarios considerando preferencias individuales.

Este trabajo de fin de máster implementa una plataforma de votación que permite a los usuarios de un entorno inteligente expresar sus preferencias sobre los servicios. Para ello se utiliza la tecnología blockchain, la cual permite desplegar plataformas de comunicación segura. La funcionalidad de la plataforma es proporcionada por la implementación de un modelo de Elección Social. Este modelo proporciona métodos de votación que permiten considerar las preferencias individuales de los usuarios para obtener una representación global. Además, una ontología es definida. Ésta es utilizada por la plataforma de votación para mejorar la comprensión de la información proporcionada y facilitar así la integración con otros sistemas.

Además, se diseña e implementa una aplicación móvil. Esta aplicación puede ser utilizada por los usuarios del entorno inteligente para usar la plataforma de votación. Por último, realizar un despliegue de Internet de las Cosas y de Inteligencia Ambiental implica un considerable coste económico y de tiempo. Por esta razón, se implementa un modelo de simulación. Éste se utiliza para estudiar los métodos de votación en un caso de uso basado en la mejora del confort térmico de los ocupantes de un edificio inteligente.

Palabras clave: Blockchain, Hyperledger Sawtooth, Sistema de votación, Métodos de Elección Social, Aplicación móvil, React Native, Simulación y Modelado, Ontología

Abstract

Smart cities and smart buildings have attracted recently considerable attention. They provide smart environments considering the use of technologies such as the Internet of Things and Ambient Intelligence. Smart environments enable to improve the quality of life and well-being of the people satisfying their demands and needs.

For this, smart environments can personalise and improve their services considering opinions, choices and preferences of the users. This process is known as the empowerment of the users or 'democratisation'. In addition, smart environments provide usually shared services. Thus, they should be able to maximise the welfare of multiple users considering individual preferences.

This master thesis implements a voting platform which allows users of a smart environment to express their preferences about services. For this, blockchain technology is used. It provides secure communication platforms. The functionality of the platform is provided by the implementation of a Social Choice model. It provides voting methods which enable to consider individual preferences to obtain an overall social representation. Moreover, an ontology is defined. It is used by the voting platform in order to improve the understanding of the information provided and facilitate the integration with other systems.

In addition, a mobile application is designed and implemented. It can be employed by the users of the smart environment to use the voting platform. Finally, approaching an Internet of Things and Ambient Intelligence deployment involves a considerable economic and time cost. Because of this, a model of simulation is implemented. It is used to study voting methods in a case of use based on improving the thermal comfort of occupants in a smart building.

Keywords: Blockchain, Hyperledger Sawtooth, Voting system, Social Choice methods, Mobile application, React Native, Simulation and Modelling, Computer Ontology

Agradecimientos

Gracias a mi tutor, por acogerme en el Grupo de Sistemas Inteligentes, por ser un mentor para mí estos años. Gracias a mis compañeros de trabajo, por todo lo aprendido y disfrutado juntos, confío en que alcancéis lo mejor.

Gracias a mi compañera de viaje, por tu afecto y motivación, compartir contigo es crecer, es ser mejor, eres mi inspiración.

Gracias a mis padres, por vuestro cariño y energía incondicional, lo que soy es gracias a vosotros, deseo que siempre estéis orgullosos.

Contents

List of Figures

CHAPTER 1

Introduction

This chapter introduces the context and the motivation of the project, including a brief overview of all the different parts that are discussed. It also breaks down a series of objectives which has been carried out during the realisation of the project. Moreover, it introduces the structure of the document with an overview of each chapter.

1.1 Context and motivation

During the last years, smart cities have attracted considerable attention [1]. This is because they provide solutions in diverse domains, such as mobility, environment, economy, governance and quality of life [2]. In order to smart cities achieves their goals optimally, citizens should be involved in the process of design, implementation and configuration of the services [1].

Citizens can participate in the definition, development and improvement of smart city services by sharing information, feedback, preferences and opinions. This process is known as the empowerment of citizens or 'democratisation' [3]. Furthermore, smart cities can personalise and improve their services considering the opinions, choices or preferences of the citizens [4].

These considerations aim to meet citizens' needs and demands, enabling smart cities to improve the quality of life of citizens [2]. For this, some works [1, 3, 5, 6] proposes the idea of platforms, which enable voting or debating processes, considering citizens' opinion and preferences.

The Internet of Things (IoT) is a technology usually used in smart cities [1]. The IoT is a network of physical object that consists of sensors, software and electronics which can communicate with each other as well as with users [7]. In particular, it implements the core implementation and enables solutions to the processing and analysing of data [2]. The concepts of IoT and smart cities are so linked that some lines of research consider the use of the term 'Urban IoT' [8].

The IoT leads to new opportunities for the Information and Communication Technologies sector, such as the interconnecting the physical and virtual environments [9]. One of these opportunities is the development of Ambient Intelligence (AmI) [10], which provides smart cities sensitive, responsive and adaptive environments [11].

Smart buildings can be considered as components of smart cities [2]. A smart building uses hardware, software and sensors for different automated or not intelligent operations, such as lighting and HVAC System control [2]. For example, the energy consumption and the comfort of occupants are parameters that can be monitored and operated in smart buildings.

Some authors have described smart buildings as an interesting way to study smart

city deployments and services in a reduced and more controllable environment [8]. Similar to smart cities, AmI is considered in buildings to enable reactive and adaptive operation models [12, 13]. It enables the building to be aware of people's preferences and actions, customising requirements and improving well-being [10, 14].

AmI in both smart cities and smart buildings can use information about people to offer services that augment their quality of life [15]. In this situation, AmI systems need to be aware of users' preferences and needs. For example, an intelligent space can customise lighting and temperature based on individual preferences.

Therefore, smart environments should not only offer a good service considering one user preferences but also make a decision in an attempt to maximise the welfare of multiple users when they use shared goods and services. For this, a solution is the consideration of Social Choice (SC) theory [16, 17]. It states voting methods to combine individuals preferences into overall social preferences.

For this, these voting methods allow individuals to choose a winning option or multiple ranked winning options. In this manner, the collective opinion is reflected. This work considers the definition and implementation of a SC model which enables to apply voting methods.

As was stated before, a solution to democratise the smart cities is the use of platforms which allow citizens to express their preferences and vote. Thus, these platforms can use the information about preferences of people to apply voting methods, such as the provided by SC theory. According to this, the design and implementation of a voting platform are considered in this work.

Blockchain technology is one of the most promising technologies in the field of the electronic voting [18] and has been recently linked to infrastructures and services in smart cities [19]. This is because it provides secure communication platforms, which have interesting characteristics such as transparency, robustness, and audibility. For this reason, the voting platform is implemented using the blockchain technology.

Furthermore, smart environments are usually composed of multiple and different systems [20]. Because of this, an ontology is defined in this work. This is used in the voting platform to provide better access and interpretation of information to both humans and other systems [21, 22]. In fact, the use of an ontology is a powerful tool to establish a common ground to exchange data and to integrate different systems [23].

In addition to the platform, a client which allow users to communicate with the voting platform is required. In smart cities and smart buildings, there is a smart environment receiving data mainly from people's smartphones and from sensors [24]. They enable that the services can be autonomously adjusted in real time.

Thus, smartphones are an important part of smart environments [8]. They provide ways to interact with smart environments, such as using an IP connection over a data-link service or a connection over Bluetooth technology. In this work, a mobile application (iOS and Android) is implemented to serve as a client of the voting platform.

As was commented before, smart buildings are an interesting way to study services and deployments in smart cities. In particular, an important parameter that can be controlled in smart buildings is the occupant comfort [25]. It is influenced by multiple aspects, such as lighting, humidity, noise and temperature.

One of the most interesting aspects to study a model based on individual preferences and a voting system is the temperature. This is because the temperature is a very relevant factor in the comfort that can be 'easily' managed by controlling the Heating, Ventilation and Air Conditioning system. Moreover, the comfort temperature value varies greatly from one person to another [26]. Therefore, the case study considered in this work is a smart building which configures the temperature in a room using temperature preferences of occupants to apply voting methods.

Approaching an IoT environment to develop and experiment with different models is quite attractive. However, deploying IoT systems involve considerable economic and time costs. In addition, the system would be deployed before any assurance could be given of the validity and usefulness of the proposed service model. Because of this, the use of a simulation model provides a good approach to overcome these drawbacks and carry out research for IoT applications [27].

Multi-Agent Systems (MAS) [28] are a method of simulation and modelling which have been recently considered for the case of use proposed, occupancy thermal comfort research. They could be defined as a collection of autonomous entities (agents), which evaluate and make decisions in an environment following a set of rules [28].

MAS are used as a solution to model problems related to occupancy. In addition, intelligent or not elements, such as sensors or equipment, can also be modelled as agents. Furthermore, MAS are an outstanding solution to carry out voting systems studies [29, 30, 31]. This is because they are a suitable combination of social science and computing.

This work implements and use a MAS to study the voting methods which are applied in the blockchain voting platform. In particular, the MAS is used to study the case of use proposed: a smart building where voting methods are applied to improve the thermal comfort of occupants.

1.2 Project goals

The main purpose of this master thesis is the design and implementation of a voting platform. It allows users of smart environments (smart cities, smart buildings) to express their preferences on the configuration of services. The platform has to meet certain requirements, such as security and privacy. For this reason, the use of blockchain technology is considered.

In addition, a client which allows users to communicate with the voting platform should be provided. As solution, a mobile application (Android and iOS) is designed and implemented. Furthermore, an ontology is defined in this work. It is used in the voting platform as a common ground to exchange data and to integrate it with other systems.

Before achieving these goals, voting methods must be modelled and implemented. In particular, methods from Social Choice theory are considered. These voting methods are used in the voting platform.

Moreover, another goal of this work is to implement and use a tool of simulation to study a case of use proposed. This is based on the improvement of thermal comfort of occupants in a smart building using voting methods. It enables to evaluate the voting methods and supports the decisions relating to the implementation of the platform and the mobile application.

According to the above, this master thesis has four goals, which are:

- Definition and implementation of a voting model which provides voting methods from Social Choice theory.
- Implementation and use of a simulation tool to study a case of use in a smart building using voting methods.
- Design and implementation of a voting platform using the blockchain technology.
- Design and implementation of a mobile application that allows users of an smart environment to use the voting platform.
- As a secondary goal, an ontology is defined in this work, which is used in the voting platform.

1.3 Structure of this document

The remaining of this document (Chapter 1 is the Introduction) is structured as follows:

Chapter 2 presents a study of the state of the art. Thus, it reviews approaches, methods, and technologies of interests to carry out this project, which are related to smart environments, Social Choice theory, simulation models, blockchain technology, ontologies and mobile technology.

Chapter 3 presents and describes the Social Choice model developed in this work. For this, it considers the modelling of votes, and the definition of voting methods and evaluation methods.

Chapter 4 describes the simulation model used to study the case of use proposed. In addition, results obtained are presented.

Chapter 5 presents and describes the architecture of the voting platform implemented. In addition, the component which provides the voting application is described. Finally, the ontology proposed is presented.

Chapter 6 describes the mobile application implemented as a solution that can be used as a client of the voting platform. Thus, this chapter describes the design, the implementation, the modules required and a description of a possible deployment.

Chapter 7 presents the closing sections of the document, which are the objectives achieved, the most relevant conclusions, and future work.

Appendixes A and B approach the possible ethical, economic, social and environmental impacts of this project and a economic budget.

Appendixes C and D presents considerations of installation, use and documentation of the systems developed in this work and a description of the ontology proposed.

CHAPTER 2

State of Art

This chapter reviews approaches, methods, and technologies that have been considered as solutions in similar work. In particular, this chapter includes the following sections. First, a description of the smart environments and its relation with voting systems is presented. Second, the Social Choice theory is reviewed. Third, the relevant aspects for the simulation model are described. Fourth, the blockchain technology is approached. Fifth, the relevant aspects of ontology are presented. Finally, a review of the technology used in the client is presented.

2.1 Smart environments and voting systems

During the last years, smart cities have attracted considerable attention [1]. As a simplistic explanation, a smart city is a place where traditional networks and services are made more flexible, efficient, and sustainable by using digital information and telecommunication technologies [2].

In smart cities, information and communication technologies are intensely used in an innovative manner. This enables to improve the city's operations for the benefit of its inhabitants by providing solutions in diverse domains, such as mobility, governance and quality of life.

Citizens should be involved in the process of design, implementation and configuration of the services in smart cities [1]. Moreover, some studies [1, 3, 4, 24, 5] investigate how citizens can help transform a city into an efficient smart city through a democratic process. These studies approach the notion of empowerment of citizens and 'democratisation' of innovation and services: the citizens must be the centre of the implementation and the benefits of the smart city projects.

Thus, citizens can share information, feedback, opinions and preferences to support the definition, development and improvement of smart city services. In the information systems research field, the participation of end-users in the design and improvement of systems has often been considered as a key factor for system quality [1]. In fact, the citizen democratisation approach in smart cities is already on the European agenda [5].

Smart cities improve and create new innovative city services that ultimately aim at improving the experience and the way citizens live in the city [5]. Therefore, if a citizen empowerment approach is not considered, the only innovative part is that technology is based on reducing humans to objects that can be measured.

In this manner, they are used as inputs for the system to react according to general and predefined behaviour, producing that citizens actually become disempowered and alienated. The citizen empowerment emerges through three factors, which are transparency, flexibility, and adaptation to individuals' needs or preferences [5].

Therefore, to shift urban management toward a citizen orientation, smart cities must personalise and improve the efficiency of their services considering the choices of the citizens [4]. These considerations aim to meet citizens' needs and demands, in order to achieve

an actual and fluent interrelation among citizens, services, and infrastructure. In fact, one essential attribute of smart cities is the improvement of the quality of life, which can be measured in terms of citizens' well-being and satisfaction [2].

In addition, governments and organisations must find ways to motivate citizens to use urban services regularly. The efficiency of urban services improves when local governments have better knowledge of citizens' preferences and needs [4].

The emphasis on citizen participation in smart cities is related to the Open Government and eGovernance movements, which argues that citizens should be at the centre of the public life via the transparency of government, participation and collaboration [1]. For this, the use of information and communications technology enables and engages that citizens participate actively in decision-making processes using democratic services such as surveys and online voting [32]

All these thoughts are related to the idea that smart city services should be at the service of citizens. Moreover, one of the most important objectives of smart cities is to increase their citizens' quality of life [4]. Therefore, the opinion of citizens is essential information. However, although the technical aspects of smart cities are being studied and covered by the literature, the essential role of the end users (citizens) in the smart cities is often ignored [1].

To solve this problem, some works [1, 3, 5, 6] propose the idea of based on internet platforms. It enables voting or debating processes, which collects citizens' opinion. For example, in [5] authors propose a system, named 'Vote a lamppost', based on citizens being able to choose the level of illumination of streetlights close to them when they walk down a street.

For this, a mobile application prototype is proposed, which enable citizens to vote the suggestion up or down. If more than 50% votes up, the lamp will change state. The empowering of citizens is provided by democratic ability to control street lighting. Authors describe this voting system as an example of citizen empowerment by providing a democratic ability to control street lighting. In addition, they affirm that this approach on services originate a change of rules so they considering not only optimisation, such as reducing energy consumption but also aspects of human convenience.

A different approached is proposed in [6]. In that work, authors propose a crowd-sourcing platform, called CrowdSC, that connects citizens effectively to local government. That platform lets citizens contribute to their community's general well-being. In particular, the platform allows users to vote the state (undamaged, damaged, very damaged) of a city

public service element. Authors describe this approach as an elegant way to make cities smarter since it fosters participation among citizens letting them take part in enhancing their own environment.

Generally, the interest of smart cities is strongly linked with the rise of new information technologies such as mobile devices, semantic web and the Internet of Things [1]. In particular, the Internet of Things (IoT) is the core implementation of smart cities [2]. It provides integrated solutions which enable to process and analyse data.

The IoT [7, 2] is a network of physical objects that consists of sensors, software and electronics which have the ability to communicate with each other as well as with users. The convergence of information and communication technologies and the internet has produced rapid growth in interest and research in this technology. Smart cities are one of the applications of IoT in the urban context, which makes cities efficient and responsive. It promises to improve the performance of urban services and the quality of life of the citizens.

In fact, there are lines of research which use the term 'Urban IoT', which is specifically designed to support the smart city vision [8, 33]. It aims at exploiting the most advanced communication technologies to support added-value services for the administration of the city and for the citizens using an enormous amount and variety of data generated. In addition, 'Urban IoT' can be used to promote the actions of the local government toward the citizens and vice-versa. In this manner, the active participation of the citizens in the smart cities' processes and services is stimulated [8].

The IoT leads to new opportunities for the information and communication technologies sector, allowing new services and applications to take advantage of the interconnection of the physical and virtual environments [9]. One of these opportunities is the application of the principles of Ambient Intelligence (AmI) [10, 14, 12, 13], which deals with sensitive, responsive and adaptive environments [11].

For this, AmI technology considers ubiquitous computing devices that interact intelligently and unobtrusively with people. In this manner, AmI provides smart environments which be aware of people's actions and preferences, customising requirements and improving the well-being of people.

Smart buildings can be considered as part of the smart infrastructure or as independent components of smart cities [2, 34, 8]. Thus, they are an interesting way to study smart city services and systems in a reduced and controlled environment where a better knowledge of the social utility and return on investment can be gained [8]. Smart buildings have different

hardware, software, sensors, and smart appliances for different automated or not operations, such as data network and lighting and Heating, Ventilation and Air Conditioning (HVAC) System control [2].

Building Management Systems (BMS) [35] are systems that control and monitor the building's mechanical and electrical equipment. For instance, BMS can control the lighting and HVAC systems, dealing with fields such as energy consumption or occupant comfort. However, traditional BMS lack of real-time acquisition systems of variable information relating to occupancy actions and preferences [36, 37]. Thus, they generally operate according to systems configured by generic and imprecise parameters.

For instance, the centralised HVAC systems are typically configured without considering different occupant preferences [38]. In some cases, systems of movement-based sensors, or similar, are considered [39]. However, these systems lack intelligent monitoring with computer support. Thus, a controlled and adaptive response is not enabled.

Similar to smart cities, AmI is an alternative to enable reactive and adaptive operation models in buildings. In this way, AmI enables BMS to be aware of people's actions and preferences, customizing requirements. Therefore, the occupants' well-being can be improved. In fact, they provide the possibility of individual room control. Therefore, aspects as occupant comfort can be effectively monitored and targeted.

A recent change in AmI is to move from a model of total and transparent automation to intelligent collaboration, in which the occupants feel that they interact and participate with the environment [40]. This is related to the interaction policy of the model proposed by J.Müller [41]. This policy enables communication between the occupants and the BMS.

In this manner, occupation preferences can be known by the building and the building can inform to the occupants about the defined configuration. A smart building with AmI interacting with occupants to improve their thermal comfort is the scenario considered in this work.

2.2 Social Choice theory

Smart environments, such as smart cities, are generally considered in physical spaces shared by various people. Thus, they should provide simultaneously suit a group of people with different preferences. In this situation, agreement technologies [42] can be used. They enable to reach agreements automatically between multiple parties applying some knowledge about

them.

Agreement technologies have covered a large variety of negotiation aspects, such as multi-issue negotiations, concurrent negotiations, or voting. A classification of the most relevant approaches in agreement technologies is proposed in [15], which is presented below:

- Auction theory: it proposes protocols and agents' strategies in auctions, in which the auctioneer wants to sell an item and get the highest possible payment.
- Negotiation theory: it defines an agreement model as a sequential game where agents alternate in making offers according to a protocol.
- Contracting theory: it proposes a protocol which allows a contractor agent to contract one or more participant agents to undertake some task.
- Social Choice theory: it considers individual preferences or interests to reach a collective decision or social welfare in some sense.

Among these options, the use of Social Choice is the most suitable option for resolving conflicts in an smart environment [15]. This is because its primary goal is to make a group decision and it is focused on maximizing social welfare. Moreover, the use of methods coming from Social Choice theory enables to maximise the comfort of the people in smart environments [13].

The theory of Social Choice (SC) [16, 17] states voting methods, which allow individuals to choose a winning option or multiple ranked winning options. For this, the voters are mapped to their preferences defining profiles, rankings of the set of alternatives. In this manner, the collective opinion can be reflected using the information of the individual preferences.

These considerations enable to apply Social Wellfare Functions [43]. These functions use SC theory over the preference profiles in order to achieve the welfare of all the community. For this, Social Wellfare Functions can know which profile has which preferences, this is, they are not anonymous. It can increase the flexibility and potential of some SC methods. For example, the votes can be weighted differently for each voter. In this way, the weight of majorities can sometimes be reduced so that minorities are occasionally better represented.

Unfortunately, there are complications to ensure theoretically that a method from SC theory enables to achieve the representativeness and welfare of all the community. In

particular, the problem outstanding in literature is the Theorem of impossibility. Nobel laureate Kenneth Arrow proved it in his most important work [44].

This theorem states that let a voting system with more than one voter and at least three alternatives, and considering certain minimal plausible fairness conditions. Then, there is not any SC method which enables to aggregate and to convert the preferences of each voter into a perfect community-wide representation [45]. For a further description of this theorem and its current implications, these works are recommended [16, 43].

Nevertheless, since the publication of K. Arrow theorem, there is a growing line of published works on SC theories and methods [46]. These works do not focus on a theoretical approach to finding a mathematical solution but on selecting SC functions which provide the most representative and suitable results. Therefore, they are focused on discovering an empirical criterion of choice among the existing SC methods [29].

In addition, both the votes and the voting results are affected by various factors, such as the voting context and the voting system [29]. Moreover, several authors report that the use of the two different SC methods can easily declare different winners [47]. Thus, the application of a different voting system to the same problem is useful to find the most appropriate solution method in similar cases [48].

Modelling and computing simulation is widely used in voting systems analysis [49, 50, 51, 52, 29, 53, 30, 31]. In fact, there is a research area known as Computational Social Choice (CSC), which is growing recently [54]. CSC applies computational paradigms and techniques to enable better use and analysis of SC methods [55]. In addition, new voting methods can be constructed.

CSC applies methods and models of multi-criteria decision-making [47], computer science, machine learning and artificial intelligent [56, 54]. In particular, CSC has been recently related to the study of collective choice in computational Multi-Agent Systems [57, 47, 58, 55, 12]. This is because they are a suitable combination of social science and computing.

Variety of real-world uses have been supported by CSC, such as web-page ranking, shared services, group recommendation systems and resources management [56, 54, 58]. Moreover, various case studies published are related to the efficient assignment of resources or shared objects to agents considering their preferences [59, 60].

In addition, there is a line of published works using CSC in order to discover an empirical criterion of choice among different alternatives of SC methods for a particular scenario [15,

29, 46]. However, there is a lack of works on evaluating these methods in new specific domains, such as an smart environment [15].

Furthermore, various authors [56, 54, 59] consider that there are tremendous opportunities concerning social interaction for the application of SC methods in smart environments. This is because new networked communication technologies, such as IoT, allow users to take social decisions with lower stakes but with higher frequency.

It provides an ideal scenario for applying methods coming from SC theory [55]. In fact, the accessing of shared resources in smart environments has not been explored in literature [15]. Moreover, there is a lack of tools to assess methods from SC theory and to decide which is the best suited in a specific smart environment [15].

According to that, this work proposes the use of SC theory to provide simultaneously suit a group of people with different preferences in an smart environment with shared services. In particular, voting methods from SC theory are studied in a case of use which considers a smart building.

Over the past decade, many efforts have focused on increasing the occupant thermal comfort using different approaches, such as Multi-Agent Systems [37, 61]. These works consider the occupants' preferences for defining the temperature in the rooms of the smart building. However, the application of an agreement technology such SC theory is not considered.

As was stated before, Multi-Agent Systems has been recently proposed to the study of collective choice. Therefore, this work proposes the use of Multi-Agent Systems to study the improvement of occupant thermal comfort in a smart building using SC theory. In particular, occupants can use methods from SC theory to vote for the desired temperature in a room.

2.3 Simulation Model

Approaching an IoT environment in a smart building to develop and experiment with different services and models is quite attractive. However, deploying IoT systems involves considerable economic and time costs. In addition, the system would be deployed before any assurance could be given of the validity and usefulness of the proposed service model.

Because of this, the use of a simulation model provides a good approach to overcome

these drawbacks and carry out research for IoT applications [27]. Modelling and simulation enable to understanding a system's behaviour without actually testing the system in the physical world. Simulation can support experimentation that occurs totally in software, although actual configuration parameters can be considered as input data.

The National Science Foundation Report on 'Simulation-based Engineering Science' [62] stated the following facts about the potential of using modelling and simulation technology to revolutionise the engineering science: (i) simulations are generally cheaper and safer, (ii) simulations can often be even more realistic than traditional experiments, as they allow the free configuration of environment parameters, (iii) simulations are often conducted faster than real time, which enables to use them for efficient if-then-else analyses of different alternatives, and (iv) simulations enable to set up a coherent synthetic environment which integrates simulated systems with prototypical components.

In this work, a case of study in which people in a smart building can vote the temperature of a room is proposed. In order to model and simulate this scenario, occupancy simulation models and thermal models are considered. This section reviews the background on these models. The aspects concerning the voting model are reviewed in Section 2.2.

2.3.1 Occupancy simulation models

The definition of an appropriate building occupancy model is necessary to carry out studies about aspects influenced by the occupants, such as occupant comfort. According to [63], four different levels of occupancy modelling can be distinguished: (i) *building level and number of occupants*, which considers the number of occupants in the building; (ii) *space level and occupied status*, which models the occupied or unoccupied state of a specific place in the building; (iii) *space level and number of occupants*, which focuses on the number of occupants in a specific place in the building; and (iv) *occupant level*, which provides an occupation model at individual occupant level.

The study of the comfort of the occupants requires to know the individual preferences of the occupants and their location in the building. Because of this, the modelling levels of building level and number of occupants, space level and occupied status, and space level and number of occupants do not consider enough information about occupancy. Thus, the model of occupant level is considered in this study to model the occupancy presence in buildings.

Furthermore, these models of occupancy can be implemented using different modelling

methods of presence and occupant behaviour. According to [64], these modelling methods can be categorized into four areas. First, *the statistical analysis*, which is used to perform studies based on probabilities considering the relationship between occupancy behaviour and environmental parameters. Second, *data mining*, which is the process of obtaining knowledge using data obtained through physical systems of presence and monitoring. Third, *stochastic modelling*, which is based on the use of Markov Chains to represent the random nature of occupant behaviour. Finally, *agent-based modelling*, which is a computational model for the simulation of agents interacting with each other and with the environment following regulated rules.

These modelling methods are characterised as following [64]. First, *the statistical analysis* is a commonly used method due to its simplicity, but it has limitations modelling accurate occupants' behaviours. Second, although *data mining* shows strong predictive capacity, it is usually limited to passive learning. Third, *stochastic modelling* achieves an accurate representation of the crowds' behaviour in buildings since a randomness factor is considered. However, this method is more suitable for long-term occupancy schedule prediction.

Finally, *agent-based modelling* is considered as a powerful technology when the problem can be understood in terms of cognitive and social concepts such as beliefs, goals, interactions, plans, roles, and norms [65]. Moreover, agent-based modelling enables to perform complex 'if-then' rules as the agents in the system can interact and change behaviours. Thus, alternative scenarios can be modelled. However, it is still at its development age methodology and requires knowing some real data about the behaviour of the occupants.

In accordance with the above, stochastic modelling together with Multi-Agent Systems (MAS) have been used in recent research lines to achieve a better representation of the behaviour of the occupation in simulations [66, 67, 68]. The advantage of this method is to exploit the potential of agent-based modelling in social modelling together with stochastic models representing a randomness factor.

Moreover, MAS have been proposed for simulating individual cognitive processes and behaviour, as well as for the exploration of social or collective behaviours. Therefore, they are used as a solution for simulating models of complex occupancy-related problems. In addition, intelligent or not elements, such as sensors or equipment, can also be modelled as agents interacting with the occupants.

In MAS, each agent assesses individually its situation and makes decisions based on the basis of a set of rules. They may execute various behaviours appropriate for the system they represent. In particular, MAS are one of the most considered methods for occupancy

comfort research [37, 69].

Furthermore, modelling and simulation using MAS is an outstanding option in voting systems studies [29, 30, 31]. This is because they are a suitable combination of social science and computing. Thus, the use of MAS and stochastic modelling is considered in the simulation model proposed in this work. In addition, information about occupants' behaviours and schedules is obtained by a survey. This information is used as simulation input data.

2.3.2 Heating, Ventilation and Air Conditioning systems modelling

Nowadays, the modelling of the physical system of Heating, Ventilation and Air Conditioning (HVAC) systems is one of the most relevant issues, which is characterised by its high complexity [70]. In fact, this is influenced by many factors, such as building materials, architecture and occupancy behaviour. Different modelling approaches have been recently proposed in the literature. According to the reviews [71, 72, 73], these methods can be classified as follows.

- *Engineering methods*, which use physical principles to model the operation of the building in a detailed or simplified way. Furthermore, these methods are based on calculating the consumption functions step by step considering the physical models of energy transfer. Engineering methods are highlighted by its precision and flexibility in modelling. However, they require a precise description of the building's physical structure and associated mechanical systems.

- *Data-driven methods*, which use prediction models based on collecting data using physical performance. Thus, these are empirical models which train using historical performance. Data-driven methods obtain results almost as accurate as the engineering methods, but they are very complex and require a high amount of historical data.

- *Gray models*, which enable to perform analysis knowing only partial data by employing a combination of the previous methodologies. These models are the easiest to develop but they are imprecise and lack flexibility.

Engineering methods have been applied in other similar works, such as [37, 69, 74, 75]. This is because these methods do not require intrinsic assumption [76]. Therefore, they

simplify load calculation providing clarity and modularity [74]. In addition, more flexibility is provided to the model, enabling to define new operating ways.

Therefore, *engineering methods* are used in this work. In particular, the *thermal zones model* is considered to model the HVAC system. This model uses heat balance methods, which is an application of the governing laws of thermodynamics.

Several works [71, 72] have highlighted the potential of tools based on *engineering methods*, such as TRNSYS [77] or EnergyPlus [78]. They enable to performing precise simulations of the HVAC system. Nevertheless, occupancy is considered in these tools [79]. Thus, some studies propose to combine these tools with auxiliary models [80, 81]. However, this option presents limitations, such as considering just Boolean occupancy (occupied or unoccupied).

Related to that, MAS have been employed recently in software for occupancy and HVAC system modelling [37, 69, 75]. As previously stated, the use of MAS provide a better representation of the occupation's behaviour. In addition, MAS enable to model the interaction between occupants and appliances. Thus, the application of engineering methods to model the HVAC system in a MAS is considered in this work. It enables to study the improvement of thermal comfort of the occupancy.

2.4 Blockchain technology

Blockchain's birth is attributed to a group or individual under the pseudonym of Satoshi Nakamoto. He published in 2008 the work 'Bitcoin: A Peer-to-Peer Electronic Cash System' [82], which describes a novel digital currency system called Bitcoin. It is based on blockchain technology, which was a solution proposed to control the order of money transactions and solve the problem of double-spending.

Since then, more than a decade ago, blockchain technology has been widely considered to revolutionize many fields beyond the financial sector [83]. Moreover, it has attracted a lot of attention and is being thoroughly researched during the last years [18]. In fact, many different and novel applications have been proposed by various researchers [83, 84].

Blockchain is triggering the start of a new era on the Internet and the online services [85]. It enables many administrative operations, fintech procedures and everyday services that can only be done offline or in person can be now safely moved to the internet as online services. In particular, one of the very promising applications of research on the blockchain is the electronic voting [18].

The characteristics of the design of blockchain architecture and its working provide interesting properties related to transparency, robustness, audibility, and security. In fact, multiple companies are high investing in this technology because they recognise the potential of the decentralised architectures so it minimises the transaction costs and makes their inherently safer, transparent and in some cases faster [86].

2.4.1 Blockchain theory and arquitecture

The theory on which Blockchain is based is simple: a system of distributed ledgers stored in a chain of blocks, which are sequentially related, and an algorithm that in a collective way negotiates and validates the content of these blocks in a network of distributed peer-to-peer nodes [87]. Blockchain can be considered a distributed database organised as a list of ordered blocks, which are usually immutable [86]. Therefore, it should be considered as a distributed append-only timestamped data structure.

Blockchain provides a distributed peer-to-peer network where is not needed a trusted authority so the non-trusting members can interect among them in a verifiable way. For this, interconnected mechanisms are used [86], which are showed in Figure 2.1 and described below.

- Transactions, which represents an agreement between two participants. They are generally the results of apply smart contracts and involves the transfer of physical or digital assets, the completion of a task or information transmissions. Functionally, transactions can be seen as a change of state of the blockchain. A set of transactions can form a block.

- Consensus layer, which solving complicated computational process, like finding patterned hashes, to ensure authentication and verifiability. Different consensus mechanisms exist depending on the type of the blockchain. Although the most outstanding is the Proof-of-Work (PoW), other protocols have recently been considered with specific objectives. For instance, Proof-of-Stake (PoS) has been used to require less power consumption and provide scalability.

- Compute interface, which enables to provide more complex functionality. A blockchain stores transactions made by the members to form a state. It is a record of dynamic variables, such as an amount of money. However, advanced applications which require not only save a state but also apply operations over the variables are implemented

using the compute interface. Usually, they are specified using smart contracts, also known as chain codes.

- Governance layer, which represents the human interactions which takes place in the physical world. The blockchain protocols are influenced by inputs from groups who integrate new procedures, methods and patch in the system. This layer deals with external-chain social processes. Thus, this process are related to the blockchain governance.

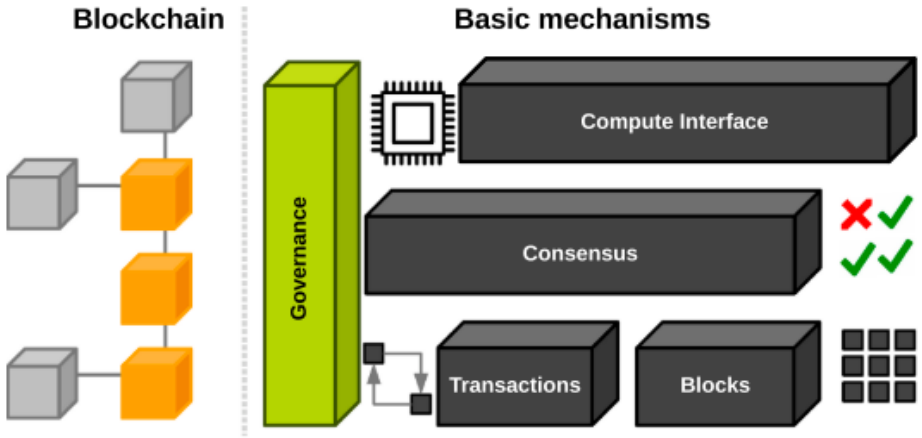

Figure 2.1: Main mechanisms or layers of blockchain technology [86]

The compute interface layer is the most interesting from the point of view of applications and services implementation. This is because it provides power to the blockchain technology by means of the smart contracts o chancoides [86]. They enable to evolve the use of blockchain in other fields beyond cryptocurrencies.

Smart contracts [85] can be defined as 'a computerised transaction protocol that executes the terms of a contract'. They enable to translate contractual clauses or functioning of services into embeddable code. Smart contracts are meaningful pieces of code, scripts or software programs. They are integrated and executed in the blockchain in a decentralised manner with minim external participation and risks.

Smart contracts can work properly, autonomously and transparently forever, being nor manipulated or removed legally once written [85]. Thus, they can be seen as an agreement and guarantee between parties who may not trust each other. In this manner, trust in a centralised authority is not required.

Recently, blockchain-based systems supporting more complex smart contracts which

implements more functionalities and interactions, enabling to establish new paradigms with practically limitless applications [85].

A smart contract is essentially the business logic running on a blockchain. It processes the requests from members of the blockchain, which contains inputs, and produces outputs. Inputs include information about the contract identifies, the transactions requests, dependencies, the current state of the ledger and values for the smart contract variables.

Then, a contract interpreter uses the inputs and the smart contract code to check the validity of the transaction and resolve it. In this manner, the appropriate outputs are generated. These are essentially a new state which is saved in the blockchain. The transactions are atomic, which ensures integrity.

2.4.2 Blockchain operation

The level of compute interface provides power to the blockchain since it enables to implements new services beyond cryptocurrency. However, this layer is not required in blockchain technology. Essentially, blockchain technology is a combination of two elements [18]: a data structure (layer of transactions and blocks) and an algorithm (layer of consensus).

Data is stored in blocks, which are units sequentially connected to each other. In this way, the blocks are organised in a chain structure. In addition to data, the blocks also store headers. These headers mainly store a reference to the previous block, information about when the block was created and a reference to the transaction data.

The references between blocks and between transaction header and transaction data are obtained by applying hash functions [18]. These functions are one-way functions. They map uniquely data of variable size with a string of bits of fixed size (hash). In addition, the hash functions produce an avalanche effect, so that any slight change in the input data produces a strong change in the hash value.

Figure 2.2 presents a simplified architecture of a blockchain. This figure shows how the blocks are organised sequentially in a blockchain. The hash reference between a new block and its immediately previous block, and the hash reference between the header of a transaction and the data of that transaction are drawn with ovals. Moreover, arrows are used to reference the hashed elements.

The blockchain algorithm defines a set of sequenced instructions. These instructions have the objective of negotiating the information that is stored in the Blockchain. The data

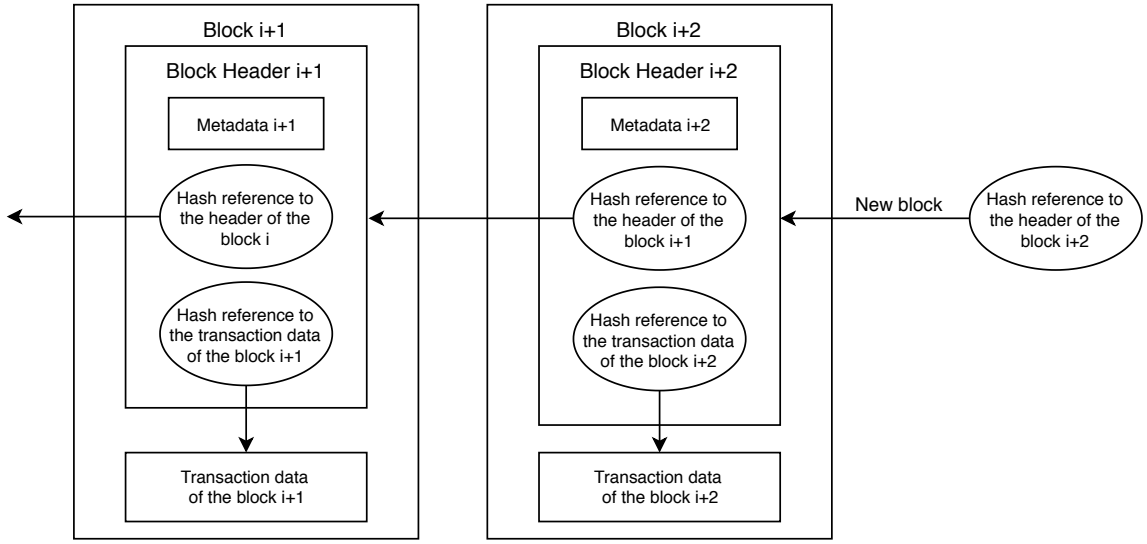

Figure 2.2: Arquitecture of a blockchain [18]

are recorded in a distributed way in each replicated blockchain of the network of peer-to-peer nodes. In this way, each node has the same version of the blockchain. All nodes can participate in a collective verification of each new block. All nodes of the network must use the same verification algorithm.

Therefore, the algorithm enables that the blockchain operates in a decentralised way without needing trusted third parties verifying. In order to a new block to be added to the chain, a number of nodes greater than 50% of the total number of nodes must give their consent [88]. In this way, a malicious node can not alter the blockchain.

For this, the consensus layer applies algorithms such as Byzantine Fault Tolerance. Typically, any participant of the blockchain is called a node. However, there are full nodes which apply the rules of validation over the possible transaction and group the transaction into blocks [86].

Besides the hashes, blockchain technology usually uses another cryptographic technology in the transactions, which is the asymmetric cryptography [88]. This technology is based on the use of two keys, the public key, which is publicly available, and the private key, which is a secret.

This public-private key pair works as follows. One side uses a key to encrypt a message so that the other side needs the other key to access the information. In this manner, two interesting operations can be applied [18]:

- Confidentiality or public-to-private. The public key is used to encrypt the message

and the private key to decrypt it. In this manner, the content of the message is a secret which can only be accessed by the owner of the private key. This operation is similar to an e-mail communication: anybody know the address and can send a message, but only the recipient can open it. In addition, this method makes it possible to demonstrate to whom the message was addressed.

- Ownership or private-to-public. The private key is used to encrypt the message and anybody with the public key can access it. In this case, the author of the message is signing it since only the true author knows the private key. This is a way of proving ownership.

The asymmetric cryptography (public-private key pair) is used in Blockchain to identify participants, usually known as accounts. For this, each account has both a private key and a public key named address. When a sender account initiates a transaction, it encrypts the transaction with the address of the receiver. Therefore, only the transaction receiver can decrypt the transaction and access to the encrypted data.

In addition, a message can be encrypted by the sender using his private key. Then, the message is broadcasting and the receivers can decrypt it using the address (public key) of the sender. In this way, the receivers can check the authorisation of the sender account.

The combination of the technologies used in blockchain provides many advantages over traditional storage systems [18]. It is very secure and resilient on both transaction and system level. Because of there is no single-centralized authority controlling the blockchain, it is a censorship-proof.

All the data is transparent as the transactions are conducted, verified and added instantly to the blockchain. In addition, there is a copy of the blockchain in all nodes of the system. Usually (although depending on the system), the data can be obtained in real time.

2.4.3 Permissioned blockchain: Hyperledger project

Two categories of blockchain can be differentiated in the literature: permissionless and permissioned [86]. The permissionless blockchains allow anyone to join as new user or node, so they are public. Thus, anyone can be a participant and performs operations, running transactions and develops smart contracts. In contrast, the permisisoned blockchains do not allow anyone to be a new member, they are privates.

In permisiones blockchain, there is usually a whitelist of allowed users who have different characteristics and permissions over the network operation. New members should be pre-verified, proving their identity. Thus, a new member needs to meet certain requirements to perform certain actions.

There are various projects belonging to both permisionless and permisioned blokchains categories. For instance, outstanding implementations of permissionless blockchain are Bitcoin [82] and Etherium [89]. In general, permissionless blockchains are considered in most cryptocurrencies developments. Moreover, the mining operation is usually required for the consensus algorithm. On the other hand, most outstanding permisisoned blockchain technology is the Hyperledger project [90]. It has developed a cross-industry permission-based blockchain frameworks.

Because of the intrinsic characteristics of private networks, they have interesting advantages. Permisisoned blockchain provides light algorithm of consensus, centralised organization, identification of users and more trust, high efficiency, low energy requirements and minimum transaction approval time (milliseconds in contrast with minutes of public blockchains). Due to this, the use of a permisisoned blockchain technology is considered in this project. In particular, a framework of the Hyperledger project is used.

Hyperledger project [91, 92, 93] is officially defined as 'a collaborative effort created to advance blockchain technology by identifying and addressing important features for a cross-industry open standard for distributed ledgers that can transform the way business transactions are conducted globally'. This project born in 2015 because different companies that were interested in blockchain technology want to work in develop a common project.

The result of that project is an open-source modular blockchain technology which helping blockchain to become a more popular, developed and industry-standard technology. For this, the goal is to create an enterprise-grade, open-source distributed framework and code base, which is supported by a technical community.

Furthermore, Hyperledger is host by Linux Foundation, which has originated a rapid grown in last few years. In fact, Hyperledger serves as a greenhouse bringing together users, developers and vendors of different markets and sectors. Because of this, it applies a design philosophy based on modularity, pluggability, high security, interoperability, API providing and cryptocurrency-agnosticism.

Hyperledger project provides a set of blockchain technologies, such as distributed ledger frameworks, smart contract engines and client libraries. This is because requirements for

blockchain vary greatly across different business applications, so there are not a one fittest all solutions [93]. The most interesting technologies from the point of view of development are the frameworks and tools. They are described as below [91, 92].

- Hyperledger Fabric. A platform for building distributed ledger solutions with a modular architecture, which provides a high degree of flexibility and adaptability.
- Hyperledger Burrow. A modular client with a permissioned smart contract interpreter following the Etherium Virtual Machine.
- Hyperledger Indy. A distributed ledger with tools, libraries and components for managing decentralized identity.
- Hyperledger Iroha. A simplified and easy blockchain framework designed to be simple and easy into projects requiring distributed ledgers. It is focused on the development of mobile applications.
- Hyperledger Sawtooth. A modular platform for building, deploying and running distributed ledgers in lightweight systems. For this, a new type of consensus (PoET) which consumes fewer resources in use. It is the Internet of Things (IoT) friendly. In addition, Hyperledger Sawtooth enables to manage the privacy of the transactions.
- Hyperledger Caliper. A tool that measures the performance of the blockchain considering a set of predefined use cases.
- Hyperledger Cello. A set of tools to bring the on-demand deployment model to the blockchain ecosystem (Blockchain as a Service). It supports Hyperledger Fabric and will support Hyperledger Sawtooth.
- Hyperledger Composer. Toolset to make it simple and fast to create smart contract and blockchain applications to solve business problems. For this, it considers assets and transactions definition. Assets are defined with a modelling language and transactions using Javascript. Hyperledger composer supports Hyperledger Fabric.
- Hyperledger Explorer. A tool that provides a dashboard for viewing data on the network, such as information about blocks, logs, statistics, smart contracts and transactions.
- Hyperledger Quilt. A set of tools that offer interoperability between different distributed and non-distributed ledgers.

As was previously mentioned, fundamental elements to implement functionality in blockchain are the smart contracts. They are essentially applications running on a blockchain and can be written in different languages. In the Hyperledger project, four frameworks support smart contracts. Moreover, each of these frameworks enables to implement them in a different language, which are [92]: Burrow in Solidity; Fabric in Golang and Javascript; Iroha in C++, C and Scala; and Sawtooth in Python, Javascript and Golang.

According to above, Sawtooth is the framework of Hyperledger project considered to implement the voting platform. This is because Sawtooth provides support for systems with limited computing resources. In fact, this framework is outstanding to be used in IoT applications. In addition, Sawtooth provides mechanisms for managing the privacy of transactions. Moreover, it enables implementing smart contracts using Python, which is the language used to implement the simulation tool and the Social Choice model.

2.4.4 Hyperledger sawtooth

Hyperledger Sawtooth [91, 92, 94] is a framework for building, deploying, and running enterprise-grade distributed ledgers. In fitting with an enterprise focus, Sawtooth is highly modular and enable to make smart contracts safe and adaptable to specific enterprise applications.

In addition, Sawtooth is designed to provide high scalability (mainly because it use of PoET consensus algorithm). It enables to manage a large and dynamic population. Moreover, Sawtooth provides low power consumption; definition of private transactions and confidential information and identity; and security against malicious actors inside the network.

Furthermore, one of the most interesting features of Sawtooth is that simplifies blockchain application development. This is because it provides a clear separation between the application or domain level and the system kernel level or core. In this manner, an abstraction of both smart contracts and transactions is provided. Thus, each developer can make his own design decisions. It enables that multiple types of applications are in the same instance of the blockchain.

Sawtooth enable to specify transaction families. They are the smart contract in Sawtooth. Transaction families can be seen as fixed transaction semantics. They limit the enabled operations of an application inside a network. In this manner, they allow developers to configure the level of versatility and risk for their network depending on the business

logic. Moreover, transaction families enable developers to write smart contract logic in the language of their choosing.

The use of transaction families provide high availability since they are independent of the network: a critical bug in a transaction family only kill this transaction family, while the rest of the network continues working. In Sawtooth, transaction families are supported by an API with a few operations like getting the state to load something from the ledger and setting the state to store something in the ledger.

In addition to this, sawtooth enables to create events associated with transaction families. It informs the subscribers of changes of state in the blockchain. In particular, three types of events are supported: notifications of possible (with or without errors) changes of state before their application in the blockchain, notifications of creation of new blocks, and specific events defined in a transaction family.

Figure 2.3 presents the official diagram of a high-level view of Sawtooth architecture [94, 95]. As can be seen, it has eight core components, which are: (i) the validator, (ii) the consensus engine, (iii) the block management, (iv) the state, (v) the P2P network, (vi) the REST API, (vii) the transaction handling, and (viii) the transaction processors. These components are described below.

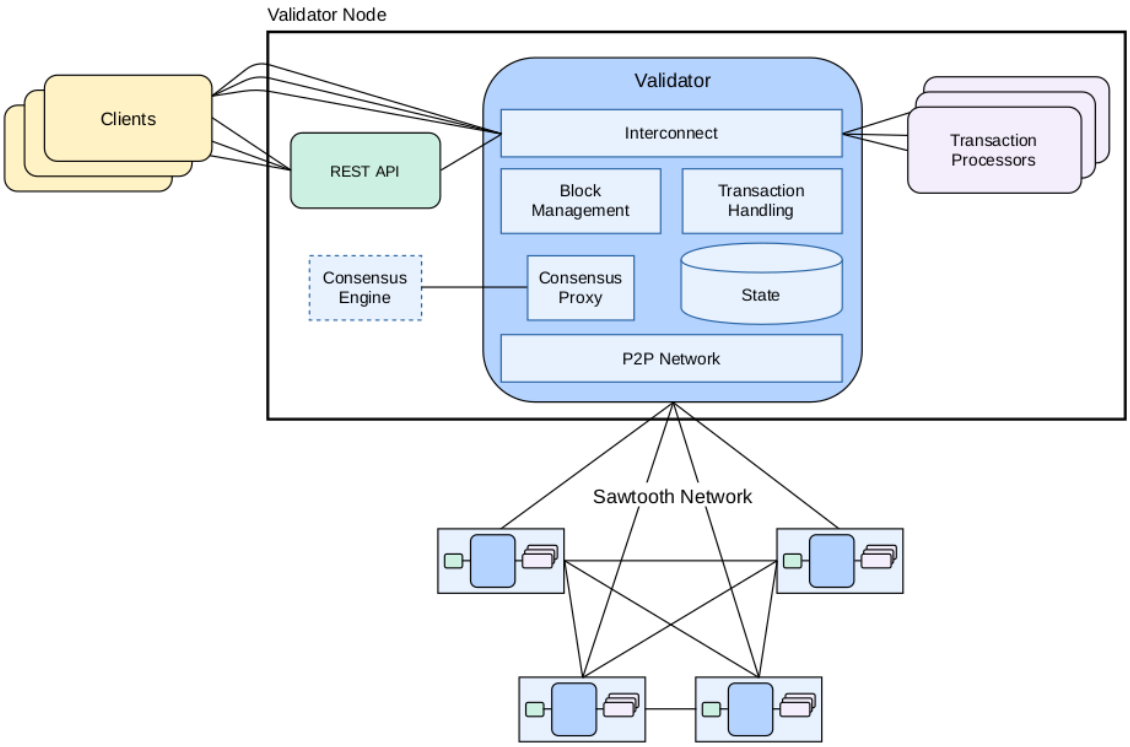

Figure 2.3: Hyperledger Sawtooth architecture [95]

Transactions are created and applied to originate a change of state. These transactions are originated by clients and submitted to the validator. The validator is the component responsible for validating a transaction and applies a change of state if the transaction is valid. In addition, the validator regulates the consensus in the network and coordinates communication among the transition processors, the clients and the other validation nodes of the network.

The consensus engine or algorithm enables nodes in the network achieves consensus on the ordering of transactions and the resulting state. Generally, Proof of Elapsed Time and Byzantine Fault Tolerance algorithms are applied. In addition, a light algorithm implemented for the development phase is provided by Sawtooth.

The distributed block management contains and operates the log, which is an ordered list of transactions in a consistent order. This component is the blockchain database, which is formed by the linked blocks. Each validator stores locally a copy of the blockchain database, which must be correctly replicated across the entire network.

The state stores the resulting distributed state after processing transactions locally in each validator and across nodes. Sawtooth stores the state serialised of each transaction processor in a single instance of an addressable Merkle-Radix Tree. It is a 'copy-on-write data structure that stores successive nodes hashes from leaf-to-root upon any change to the tree'. Each transaction processor is allocated by a namespace. Moreover, serialization methods are left up to the application developer. These enable to define the rules for serializing/deserializing the data.

The peer-to-peer network enables to pass messages and transactions between nodes. These messages are communicated over TCP and contain information about transactions, blocks, peers, and more. In addition, through access control lists, Sawtooth nodes control who can connect to the network, who can participate in the consensus process and who can submit transactions.

In addition, Sawtooth provides a REST API. It enables to interact with a validator using HTTP/JSON standard in order to submit a transaction and read blocks. Mainly, it is defined to be used by a client. The operations of authorisation, such as signature verification, are made by the transaction processor. Therefore, the REST API simply submit a transaction to the validator, which forwards them to the corresponding transaction processor, and fetches data stored in the blockchain.

The transaction handling works like a router. It receives transactions from clients and

filters them, validates them and forwards them to the transaction processor specified in the transaction. One or more transactions are grouped into clusters, which represents an atomic unit of state change. Transactions and clusters have a payload, which defines the change of state, and a header.

Serialisation operations are applied over both fields. Only the transaction family which process the transactions deserialize the payload, to the rest of components of the systems and transaction families the payload is just a sequence of bytes. In addition, the payload is encrypted with SHA-512 and the headers of the transaction and clusters are signed with private keys. These signatures are matched with public keys, which are included in the headers.

The transaction processors specify the applications in Sawtooth. They validate transactions and originate change of state. For this, a transaction processor applies the rules that have been established in the associated transaction family. A transaction family is the definition of business logic. Thus, it defines the taxonomy of available operations or transactions that can be applied in the blockchain. In addition, a transaction family implements a data model, which will be used to store the changes of state in the blockchain.

The transaction processors and transaction families work in Sawtooth as state machines and smart contracts. They processing the content of the transactions as input to produce an output. Thus, these components implement the business logic in the same way that a smart contract. In addition, they can be seen as state machines since they use a new transaction with a snapshot of the current state to generate a new state. Then, the current state is updated.

Furthermore, the transaction processors are smart contract written in a high-level language. They are compiled down to Etherium Virtual Machine bytecode. One of the drawbacks of this design is that creating a domain-specific transaction processor limits the types of actions that can be performed on the blockchain.

However, it improves security and performance. In addition, supporting more mature programming languages enable to use of existing tools that can perform a static, dynamic, and formal analysis of smart contract code.

For the development, Sawtooth provides SDKs (Software Development Kits) in Python, Javascript, Go, C++ and Java. In this project, the Python SDK of Sawtooth is used to develop the voting platform based on blockchain. Moreover, the functionality of the voting application is implemented through the definition of a transaction processor and a

transaction family.

2.4.5 Blockchain and voting

Blockchain thecnology has been recently proposed to implement secure communication platforms which enable to integrate infrastructures and services in smart cities [19]. The aim is to provide traditional and new services in a decentralised and efficient way while the same degree of validity is ensured. Registration of legal documents, identification, contracts, taxes manages and voting are outstanding examples.

Related to voting, some projects are focused on ideas related to democracy, where the authority and decision-making are distributed throughout self-organising teams and individuals instead of relying on typical institutions models [86].

Voting is the fundamental basis of democracy. However, despite complex security mechanisms, traditional voting mechanisms can suffer fraud [96]. In addition, these voting systems are slow due to the vote collection mechanisms [18]. This is because the votes are usually collected from different locations for then to be centralised in a common institution.

Furthermore, voting results obtained by traditional voting mechanisms are not verifiable by voters. They cannot verify that their votes have been correctly considered and that the voting method was properly applied [97].

These problems related to traditional voting systems have led to an interest in the creation of different electronic voting (e-voting) solutions [18]. E-voting is the use of electronic means and/or information technologies to perform voting procedures. It can provide solutions to problems that affect the voting process, such as increasing the speed in result obtaining and allowing voters to know information on the voting process.

E-voting is considered a promising and inevitable development which provides interesting advantages and the development of stronger and more direct democracies [86]. In fact, the use of information and communication technology in the electoral process is continuously rising around the world [98]. E-voting is a trending, yet critical, topic related to online services.

However, e-voting systems must face challenges not only technical but also procedural. For example, although the risk of fraud is reduced, there is the possibility of hacker attacks. In addition, there is an absence of standards which can cause voters to mistrust and misunderstand the voting system.

All the e-voting systems must comply with some basic requirements in order to the system can be considered valid [18, 96]. The most important requirements are the following. First, the availability of the system, which must be accessible and usable by any voter. Second, vote integrity, which protects the votes from being modified, removed or duplicated. Third, transparency and verifiability of the voting procedure. Fourth, correctness during the voting method application. Fifth, voter identification and authentication, which check who can vote. Finally, voter privacy, which protects the secret choices of the voters, keeping it anonymous.

Moreover, one of the most important and prevalent problems is lack of auditing and transparent capabilities and system verification methods [86]. In addition, most electronic voting systems solutions have a proprietary and centralised design harm the trust and confidence of the voters.

In contrast, blockchain technology is an e-voting alternative which can meet the requirements and provide solutions to these problems [18]. This is because blockchain enables to develop a platform for public verification of information. In this manner, the information about the system and process of e-voting can be evaluated by all the voters.

Furthermore, e-voting systems based on Blockchains can provide to the voters with many advantages. They are related to voting procedures which are secure, anonymous and verifiable over an internet connection [18]. These desired properties are provided by using the cryptographic methods considered in blockchain technologies. In fact, the recognition of the potential of this technology in voting systems has originated that some countries already started researching and implementing these e-voting systems [18], such as United Kingdom, United States and New Zealand.

Some voting platforms similar to the proposed in this work have been implemented using blockchain technology. Of these, some have been implemented considering permissioned networks, such as Hyperledger. For instance, [99] and [100] have considered to use Hyperledger Composer.

In addition, it seems that only one project, Bitagora [101], has used Hyperledger Sawtooth for the implementation. However, this project seems to be no longer available and its author has acknowledged that 'it is not fully functional'. Furthermore, none of the voting projects found based on blockchain considers the use of methods from Social Choice theory.

2.5 Semantic technology: Ontology

In the field of philosophy, ontology is the study of the nature of being [102]. In computer science, ontology can be applied to clarify and define concepts, categories and relations [102]. This enables that the models of knowledge representation are committed to a conceptualisation. An ontology in computer science is the specification explicit and formal of this conceptualisation [103].

Ontology is usually used in projects related to natural language understanding, information retrieval, theoretical investigation, knowledge exchange and reuse, simulation, and modelling [104, 105, 106, 107, 22, 108]. Despite being projects with different approaches, all ontologies consider the definition of the same elements, which are terms, relations among terms, and properties of the terms [102]. Each ontology defines each element depending on the purpose it was created for [109].

Related to knowledge sharing, the modern information systems are moving from 'data processing' toward 'concept processing' [110]. This produces that basic unit of processing is transformed from a piece of data to a semantic concept. In this manner, the precision and efficiency in the evaluation of information are improved [111].

In fact, ontologies are designed to promote greater consistency in the description and understanding of the meaning of information in both textual definition (human understanding) and logical definition (computer understanding) [21, 107].

Furthermore, the use of ontologies as a mean to exchange information between heterogeneous sources is one of the most important subjects on semantic technology [112, 106]. In fact, an effective ontology is a powerful tool to establish a common ground to integrate different systems [23].

According to this, the ontology proposed in this work aims to define a set of semantic concepts, relations and properties related to voting and Social Choice theory. It is used to provides a semantic knowledge exchange system to the voting platform.

Ontology research has emphasised the importance of reusing ontologies already defined [113, 114, 115]. In this manner, an already developed ontology can be used to describe knowledge in a new ontology. The reuse of ontologies enables to focus the effort on increasing the level of detail in the representation of knowledge.

However, at the time of this project, no ontology of voting and Social Choice theory has

been found. Thus, the proposed ontology is defined with the motivation that it can be used by related future ontologies.

2.5.1 Building an ontology

In general, an ontology is formed by three main elements, which are terms, properties and relations [116, 117, 118, 119]. These three elements are described as follows. Firstly, a set of generic terms are defined in a taxonomy or lexicon. These terms represent things or facts of the scope approached. Terms are also named class, concepts, type, category or kind [120]. An instance of a term or class is an individual.

Secondly, properties are used to characterise semantic terms and to facilitate concept detection [119]. Thus, properties usually are simple, distinctive and easy to identify. In addition, each property can be defined as part of one category.

Finally, relations provide semantic links among two or more than two terms and individuals [120]. Each relation can be represented very differently in each ontology. For instance, the part-whole, which is the most common relation, can be represented in seven different forms [121].

The definition of a building ontology methodology remains a complicated and tedious task [113]. Many authors, such as [122, 113, 114, 115, 111], stand out that the integration and reuse of methodologies is very interesting for a greater collaborative and interoperability work. According to this, various research works have proposed a series of steps or phases to establish a common general methodology for developing ontologies.

Four of the most outstanding methodologies have been studied, which are presented in [111, 119, 118, 117]. The building ontology methodologies presented in these works present important similarities. This fact reinforces the validity of the approaches. In this work, the methodology proposed in [117, 123] is considered since it is the most complete. This methodology has the phases presented below.

1. Definition of the purpose and specification of the level of formality. Identify and characterise the range of intended users, the role of the ontology and the motivating scenarios (uses and mechanisms).

2. Description of the scope or subject matter. Definition of a set of concepts and terms covering the range of information described by the ontology.

3. Building of the ontology. Specification of the ontology using the set of defined terms and concepts. In this manner, definitions are produced. For this, the key concepts and relationships are captured. Moreover, the representation of the conceptualisation in a formal language is performed. In addition, the use or integration with existing ontologies is considered.

4. Evaluation or revision cycle. Application of criteria to evaluate the result of the ontology building. In particular, authors classifies the criteria in two sorts. First, general, which evaluate the clarity, consistency and re-usability. Second, specific, which check the ontology against particular aspect such as the identified propose.

5. Documentation. Definition of a guideline to facilitate the use and understanding of the ontology. This is essential to enable effective knowledge sharing.

Besides building methodologies, some authors [118, 111, 117, 124] propose considerations to design an ontology. The reviewed works propose the following considerations: realism, representing technical concepts; perspectivism and modularity; fallibilism, enabling effortless and successful changes and tracking; adequatism; precision; conciseness and clarity; and reusability. In summary, four principles are considered: lightweight, completeness, compatibility and modularity.

2.5.2 Ontology technologies

A large amount of effort has been dedicated to the development of semantic markup languages [120]. This languages enable to publish contents and codified metadata about the meaning of the content. Some important examples of such languages are the XML-based extension of HTML (XHTML), the Resource Description Framework (RDF) and the Web Ontology Language (OWL). Various works related to knowledge exchange and integration between systems have proposed the use of OWL to implement ontologies [125, 109, 126, 122, 21, 105]. This is because OWL is powerful provides interesting advantages.

OWL is an ontology language which was primarily designed by W3C to represent information about categories of objects and interrelations among them [127]. It has been influenced by established formalisms and paradigms of knowledge representations, ontologies and Semantic Web languages [109]. The most important influences are the RDF, the Description Logics and DAML + OIL.

Therefore, OWL extends RDF, so there is compatibility between both technologies.

RDF provides data model specifications and XML-based serialisation syntax. Therefore, OWL takes the basic fact-stating of RDF and the structuring capabilities of RDF Schema. In particular, RDF (and RDFS) provides OWL the basic features of classes, properties, domains, ranges, restrictions, and relations [109].

Moreover, OWL use Description Logic. This is a formalism of knowledge representation based on classes and characterised by using constructors [104]. In this manner, OWL models the knowledge with an object-oriented approach. Thus, OWL enables to declare classes organised in a hierarchy and as a logical combination of other classes.

In addition, Description Logic allows OWL to consider datatypes and values, such as integers and string [125]. DAML + OIL provides a model theory to formalise the meaning of the language. This is crucial if the ontologies exploited by intelligent agents [127].

Three different solutions based on OWL are available [128]. First, OWL-DL, which is a friendly syntax defined mainly in Description Logic manner. Second, OWL Lite, which is a similar syntax to OWL-DL but simpler. Finally, OWL Full, which is a version with better compatibility with RDF/RDFS and greater expressive power. In this work, the ontology has been implemented using OWL Full. This solution is considered in order to ensure an inter-operable, easy-to-read and powerful knowledge representation.

Regarding the format, OWL Full (as RDF) requires a definition based on triples annotating properties. [129]. In addition, it uses URI references as names. Various readable representation syntax exists for rendering OWL Full Ontologies, such as RDF/XML, Turtle and JSON-LD [130]. The ontology proposed should use a format easy to be read for both machines and humans. In this manner, readability and easy integration can be provided. Therefore, Turtle syntax and JSON-LD syntax are considered, which are less verbose than RDF/XML.

The definition of the ontology can be done using a supporting tool. This review [115] presents and compares eight of the most relevant tools for ontology development. Of these tools, Protégé stands out for being open source and enabling work with many technologies and formats (XML, Turtle, RDF(S), XSD, JSON-LD...). Moreover, Protégé is standalone and has a strong extensible architecture with plug-and-play capability. This enables that many plugins are available for multiple functionalities.

Ontologies, as a data structure for conceptualising knowledge, can be built in many different ways [110]. Thus, it is interesting to perform an evaluation of which version of the ontology is better. For this, various approaches to evaluate ontologies have been considered

in the literature.

According to the review [110], there are four main evaluation approaches. First, based on comparing the ontology to a 'golden standard'. Second, based on using the ontology in an application and evaluating the results. Third, based on performing comparisons with a source of data related to the domain to be represented by the ontology.

Finally, based on requesting humans an evaluation of predefined criteria, standards or requirements. In this work, the ontology is evaluated applying the second and fourth criteria. These are considered since there are no similar works that can be used for comparison.

To carry out experiments, various authors [105, 125, 21, 112, 131] propose to use Query Languages (QL). They enable query the information defined by ontology languages [132]. In particular, SPARQL [133] is the W3C recommendation of OWL-based QL. It enables to infer information from an OWL ontology basing on the notion of RDF triple pattern.

Furthermore, various authors [125, 109, 132, 21] propose to use Jena Fuseki Semantic Web Toolkit to build context reasoners. It provides a SPARQL engine which can be used to define queries that check entities, properties and relationships. According to this, the operation process proposed in this work to evaluate the ontology is based on three steps: i) extraction of the data from the source (voting platform), ii) transformation of the data into OWL data format in the target vocabulary (Turtle and JSON-LD) and iii) load of the OWL data into a Jena's SPARQL endpoint.

2.6 Mobile client: React Native

One of the most popular ideas about smart cities and smart buildings refers to an environment that receives data from sensors and from people's smartphones [24]. This allows the smart environment to autonomously adjust and deliver better services in real time. For this, mobile phones of people are usually just considered as resources that can provide sensory data, such as accelerometer, temperature and location, to feed the smart environments' systems in a fully anonymised or not way [5].

However, smartphones are a more important part of the IoT systems. They provide other ways to interact with the environment beyond the sensors [8]. For instance, the NFC transceiver integrated into last-generation smartphones can be used to identify tagged objects.

Furthermore, mobile devices can provide access to the IoT system in different ways [8], such as through an IP connection provided by the cellular data-link service or setting up a direct connection using short-range wireless technologies. Smartphones are part of the IoT framework in smart environments being key elements to obtain information [2]. In fact, smartphones are one of the outstanding solutions proposed in the literature to be used as clients of systems of E-voting [98].

In this work, the citizens that are expected to live in a smart city are considered digitally educated and in possession of a smartphone. Since the release of the first smartphone, the use and demand for mobile applications have increased rapidly [134]. This technology enables to use the individual opinion not only for voting for the best singer in a talent show but also for serious referendums or political elections, making the smart-phone the ultimate symbol of citizenship [24]. Thus, non-digital citizens have apparently little room and a limited voice in the city of the future [24].

An application that allows users to interact with the voting platform is implemented in this project. Creating a mobile application usually requires to develop two different versions, one for Android and one for iOS, which are the two leading operating systems for mobile devices [134].

This is because although both applications may have the same logic, they have different components of the user interface (UI), and the applications themselves need to be implemented in different languages. Thus, there is a replicated process that requires more time and knowledge costs.

To solve this problem, efforts have been recently made to develop frameworks enabling hybrid implementations or cross-platform developments [134]. One popular technique is to create an application on the web can be accessed by mobile through a browser. For this, features of HTML5 and libraries such as Bootstrap are used to implement a responsive website, which is easy to use with either a computer or a mobile device.

Another alternative is to use a hybrid framework which combines web and native development. For this, the capabilities of the devices are used by the JavaScript API. However, none of these approaches has been able to provide a native feeling of the resulting applications [134, 135].

As a solution, Facebook released a new framework called React Native, which has revolutionised the way mobile applications are created [134, 136]. Since then, many companies such as Uber, Skype or Tesla have considered this framework in the implementation of their

mobile applications.

This solution is known as cross-compiled [137] since the framework use as input an application implemented in a not-native programming language and transform it to a native code of a mobile platform.

React Native [138, 134] is a Javascript framework for writing real, natively rendering mobile applications. This technology enables to use existing knowledge of Javascript and React to build and deploy fully featured mobile applications for both iOS and Android, with 100% code reuse between the two platforms, that truly render natively.

Therefore, there is an approached 'learn one, write anywhere'. In addition, it provides plenty of advantages over traditional means of mobile development without sacrificing the native look and feel. React Native provides these and more advantages because it renders using real mobile UI components.

There are methods of writing mobile applications rendering web views. However, these present drawbacks, especially around performance. In contrast, React Native actually uses native UI elements and works separately from the main UI thread, maintaining high performance and capability. Moreover, React Native provides perks related to debugging and saving development and production time.

One of the reasons for the success in the association between the native components of the mobile platforms and the components implemented in React Native is due to one of the most important features of React Native: the virtual DOM [138]. It works as a layer between the description of how things ought to look and the actual rendering of the application onto the view.

The virtual DOM, in contrast with the browser's DOM, rerenders the minimal necessary changes, which has a significant impact on performance. It is composed of components, which are translated to the browser's DOM in React and to the mobile platform specific UI View in React Native. This process is possible because of a bridge, which relates the abstraction layer of the Virtual DOM to multiple different platforms, such as web, iOS and Android. This operation is presented in Figure 2.4

Components are the fundamental building blocks of React, which have four main concepts [136]: data, which comes from other components (properties) or from somewhere and is rendered by the component; events, code which responds to user interactions; JXS, syntax to describe UI structures; and lifecycle, methods to control the processes of change in the

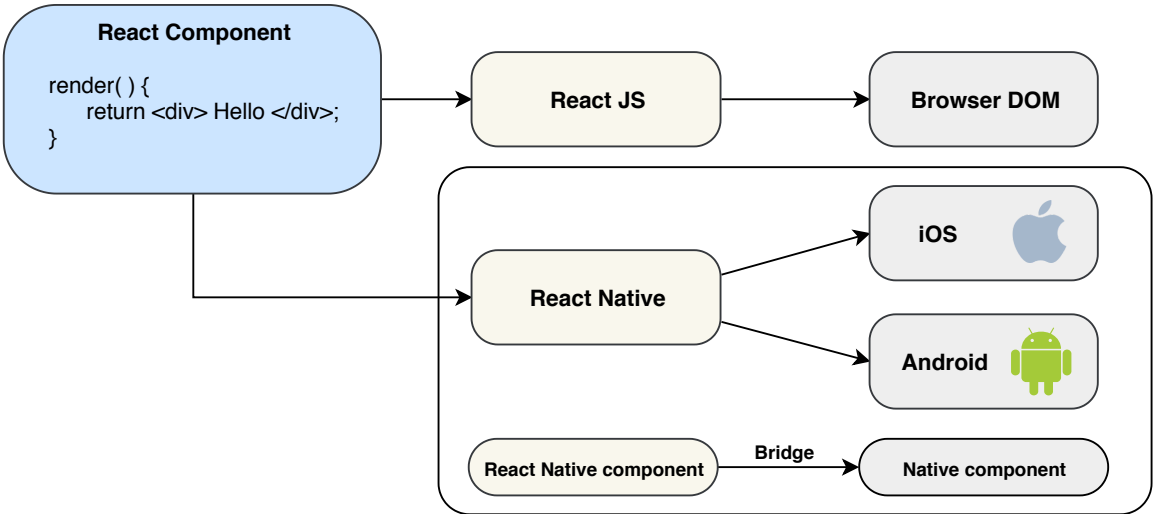

Figure 2.4: React component rendering with React JS and React Native

lifecycle of the component.

Components are the way to describe UI structures, which are declared using the vocabulary of JSX markup. Thus, the vocabulary of JSX can be extended by implementing new React components, which can have parent-child relationships and enable to share data.

The data in React Native components has two parts [134], which are the state and the properties. The state is the dynamic part of the React component. In addition, when the state change, the component is re-rendering.

The properties are the data passed by other components only when the component is rendered. They are different from the state because they do not change after the component is rendered. The properties can be also functions, which enable to manage events between father and children components. One of the most interesting aspects of the components is that they are reusable. In other words, the same implementation can serve multiple purposes. This reduces the complexity and clarity of the implementation.

React Native handles the screen providing some advantages in comparison with a web-based solution, such as the quality of interaction with the application. This is because the system handles touches in a complex view with high-level features. Moreover, gesture recognition is more advanced on a mobile device than on the web. Therefore, React Native consider different components which are adapted to multiple mobile-interaction options.

CHAPTER 3

Social Choice Model

This chapter describes the Social Choice model proposed in this project to provide voting methods. For this, it presents the modelling of votes, the definition of voting methods, the voting mechanism, and other considerations.

The modelling of voting methods from the Social Choice (SC) theory is used in this work to provide the functionality to the voting platform. It allows users to define individuals preferences, which are combined to reflect a collective opinion which improves the common welfare. Therefore, this section describes five aspects which form the SC model. These aspects are: (i) the modelling of votes, (ii) the different voting methods, (iii) the voting methods evaluation, (iv) the importance of considering randomisation and possible manipulation. Finally, the solution proposed is presented.

3.1 Votes modelling

The modelling of votes is the specification of their possible values before applying voting methods. In general, there are two types of votes [139]. First, those in which voters rank candidates. Second, those in which voters grade the candidates with binary values (approve or not approve) or non-binary values (inside a range).

Furthermore, the value of the votes when they grade the candidate with non-binary values also should be defined. One outstanding solution proposed in the literature is given in [17]. In this work, authors describe a voting system where both voters and candidates are associated with precise points in a metric space.

In this situation, each voter tends to prefer candidates that are closer to her. Therefore, it is desirable to select a candidate that minimises the sum of distances to the voters. This approach makes it easier to evaluate voters' satisfaction with the winning candidate using the value of the distance.

Nevertheless, it is difficult to establish precise spatial points representative of candidates. In addition, typically voters are unable to precisely pinpoint their position [17]. Thus, it is more realistic to expect the voters to provide their rankings of candidates or grade numerical values merely representative.

For this reason, common voting rules operate on voters' preference rankings [17]. However, this approach rules not enable to use of distances and complicate the choice of the best rule to choose the winning candidate.

According to this, the votes modelling proposed in this article considers the use of merely representative numerical values for all the votes. In this manner, both the evaluation of the SC methods and the definition of preferences by voters are easier.

However, some SC methods require non-numerical rankings. Therefore, in order to cover all cases, some transformation operations are considered. These are performed before the votes are given to the SC methods. Moreover, the votes modelling operation may be useful to some type of services. For instance, in demand-based services, where the votes of certain users can obtain preferential treatment under some conditions.

In addition to this, some authors [140, 141] consider important to allow voters to express dissatisfaction with each candidate. Therefore, it has been considered that the votes can be also negative values. In particular, a range between -10 and 10 is considered to model the votes of the voters for the preferences. This is a similar scale to the proposed in other works [142, 15, 17].

3.2 Social Choice methods definition

In the research of SC theory, several voting methods have been proposed for aggregating voters' preferences. In general, these methods can be classified into two types, similar to the classification of vote modelling. First, preferential or ordinal voting methods, which only use ordered preference of the alternatives (ranking of candidates). Second, non-preferential or utilitarian voting methods, which use graded preferences of each alternative (intensity of each candidate).

The methods belonging to the first type are Borda voting and Pairwise comparisons voting. The rest of the methods belong to the second type, considering binary or non-binary values. Note the reader that, as was commented in Sect 3.1, voters can also disapprove candidates. In particular, the SC methods considered in this work are described below. For a more detailed description of the methods used, the consultation of these works [143, 144, 15, 17] is recommended.

- Borda voting. In this method, the voters provide full ordering preferences. Therefore, let n candidates, each voter give n-1 votes for the most preferred candidate, n-2 for the second most preferred candidate, so 0 votes are given to the last preferred candidate. The alternative with the largest number of total points is the winner.
- Pairwise comparisons voting. This method applies a Borda voting and then performs a head-to-head match between each alternative and the reminder alternatives. In this manner, each alternative obtains one point for each win and half a point for each tie. The alternative with the most total points is the winner.

- Plurality voting. In this method, each voter gives a vote to just one candidate. This is, just the favourite candidate is labelled with 1, and the remaining are labelled with 0. Thus, the alternative with the most first place wins.

- Approval voting. In this method, voters approve and disapprove any number of candidates as they wish. Therefore, the votes are represented as -1, 0 or 1. The candidate receiving the greatest total number of votes is the winner.

- Single Transferable Vote. this method, which is also known as the Hare system, applies Plurality voting with elimination. After each voting round, the alternative with fewer votes is removed. Then, the Plurality scores are recalculated using new ordering preferences of each voter. The process finished when one alternative has the number of votes greater than a threshold, which usually is half of the total number of votes. This method is interesting because increase the representation of the minorities and minimises the problem of the useful vote.

- Range voting. In this method, each voter gives to each candidate a number of votes within a specified range. In particular, the range of the votes considered is -10 to 10. Then, the candidate with the largest value of votes is the winner.

- Exchange of weight voting. This method applies a Range voting and then evaluates a condition for the winner candidate. This condition is that the voters who have preferred the winner candidate have enough negotiation weight. If they have not, the next most voted candidate is evaluated following the same condition. In this manner, the minorities can obtain enough weight of negotiation after several negotiations to achieve that their preference would be the winner. Thus, this method increases the representation of minorities.

- Cumulative voting. In this method, the voters can divide a maximum of v votes among all the candidate as they prefer. In addition, a maximum value of votes m for the same candidate can be defined. Then, the candidate with the most number of votes wins.

All the modelled methods enable the possibility of a multi-winner election. In a multi-winner election, the voting rule picks a group of some K winner candidates [145]. This consideration is interesting for some usual applications, such as picking items for share among users or chosen a representative committee. In general, a value of $K = 3$ is considered in this work.

3.3 Social Choice methods evaluation

As was stated in Sect 2.2, two different SC methods can easily declare different winners. Thus, the aim of experimenting with different voting methods to approach the same problem is to find the most appropriate voting method in that case and in similar cases. Therefore, It is necessary to define a way of evaluating and comparing the results of different voting methods.

For this goal, various authors [146, 17, 147] propose the distortion as a measure of the quality of a voting method. They consider a model where voters have candidates which provide utilities and welfare. Then, the goal is to select the candidate that maximise the total utility and satisfaction [58].

The satisfaction is calculated as the sum of distances between the voters and the winning candidate. This value is considered to calculate the distortion. According to [146, 17, 147], the distortion is defined as the ratio between the satisfaction of a candidate selected by a method and that of an optimal candidate.

As the reader might expect, and in accordance with K.Arrow's theorem, some amount of distortion is unavoidable. In particular, Plurality voting, Borda voting and Approval voting, which are the most popular voting rules, have a very poor distortion value [148, 17]. In contrast, the authors out stand the methods which follow the Copeland rule [149], which require pairwise evaluations between candidates. Therefore, they consider the Single Transferable Vote rule the best option for a moderate number of the candidate.

However, according to the author who proposed the distortion [146], there is a hard computation problem associated with its calculation. This may not be a problem in long-term stable systems. However, a smart environment in constant change is considered in this project. Therefore, the calculation of distortion may not be feasible. Thus, the measure considered is the satisfaction value of voters.

The satisfaction or utility of a voting method is calculated as the distance between the preference of each voter and the winner candidate [15]. In particular, the metrics considered to evaluate each voting method are mainly the accumulated satisfaction of all users and the medium time without the wanted configuration. This second measure is useful to evaluate the problem of minorities.

3.4 Manipulation and randomisation

Voting systems have proved to be a proper and efficient tool for making group choices among alternatives [58]. In addition, all the voting systems are defined with the purpose of being fair and difficult to manipulate [47, 15].

However, it has been shown that every voting method is subject to a greater or less degree of manipulation [150, 151]. In fact, studies have proved that the best strategy of voters is to manipulate an election outcome by misrepresenting their preferences [47, 59]. For example, one strategy might be to vote for the most probable winner or vote against the least preferred alternative [58].

Related to that, the awareness of the voters on the voting system and the results is an important issue. In this situation, voters can apply "strategic behaviour", which is related to game-theoretic [55]. In this manner, concepts such as the Nash equilibrium [152] must be considered, which can produce huge difficulties.

This fact can originate infinite loops of reasoning ('I should consider that they know which I consider...'), and complex dynamical models, where voters change their votes in response to the observed outcome [59]. A recently presented voting method tries to turn this problem to a virtue. For this, this method considers that all voters know the voting system and can use this information to vote strategically. However, this method is considered unstable and unreliable [58].

Therefore, considering of manipulation or of a strategic behaviour causes complications with computing and with the fair application of voting methods. Two options are proposed as a solution in the literature [59, 55]: to consider that the voters are ever fair or to apply models based on assessing voters' integrity.

In particular, one possible model for assessing voter integrity can be the consideration of a historical record of voter behaviour, which is related to gamification. However, because of the goal of this work is to experiment with the use case and the voting methods, all voters are considered fair. In addition, this assumption simplifies greatly the model. Therefore, the consideration of unfair voters and the before commented solution is proposed as future work.

Randomisation plays an important role in the SC theory. This is because randomisation seems particularly fair and natural in situations with difficult solution [55, 59]. In particular,

two problems are approached in this model using randomisation. The first problem takes place when voting leads to a tie.

The second problem is Condorcet's paradox [153, 154]. This problem refers to situations in which collective preferences are cyclical even though individual preferences are not. This generates a paradox and produces a situation without a solution, which makes impossible to determine a winner.

Randomisation can be applied to solve this problem producing nevertheless an unstable system [155]. This solution is unacceptable in systems that require long-term stability, such as political systems. However, their application in smart environments can be considered acceptable. This is because, as was commented before, they are group decisions with lower stakes and higher frequency.

3.5 Social Choice model proposal and implementation

This project defines and implements a model to apply the voting methods from Social Choice theory. Figure 3.1 presents a activity diagram with the components considered in this model, and the input and output data. As can be seen, users' votes or preferences are used by three different components before results with the winning candidates or configuration are obtained. Furthermore, there are two components which provide historical data and satisfaction values.

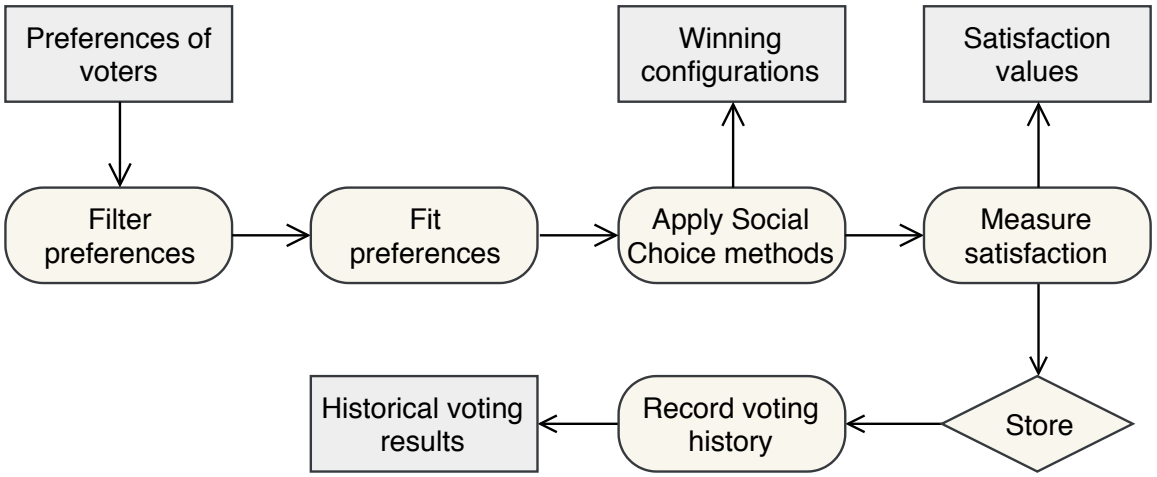

Figure 3.1: Activity diagram of the proposed model to apply voting methods from Social Choice theory

The flow of activity is as follows. Voters' preferences are used by two components

that apply operations of preferences filtering and fitting. Then, when preferences have valid values for a specific Social Choice method, it is applied. In this way, the winners of the voting are obtained. The results and votes can then be used to calculate satisfaction measures. In addition, it is possible to store a history of the voting results and the satisfaction measures obtained.

The proposed Social Choice model is implemented using the Python programming language. The implementation is done to provide a reusable and extensible package, which is named 'socepy' (**so**cial choi**ce** in **py**thon). Of course, this is available for free and in open source. Figure 3.1 presents a class diagram with seven components that constitute the implementation, each of these seven components is described below.

- Execution. This component receives Preferences of voters and returns Winning configurations and Satisfaction values. It can be used in two ways, considering storing historical values using the Voting history record component or not.

- Filtering of preferences. This component filters and adjusts the values of the preferences so that they be inside the limits and allowed values of the model. These allowed values can be modified. In addition, the filtering operation ensures that the format of the votes is the appropriate one, verifying for example that each configuration is voted only once.

- Social Choice methods. This component contains all considered Social Choice methods. In particular, eight voting methods are considered, which are described in Section 3.2. To apply these methods, support methods from Fitting of preferences and Basic operations components are used.

- Fitting of preferences. Methods for adjusting preference values are provided by this component. They are used to adapt the generic format of the preferences to the specific format of each Social Choice method. This is an essential operation since some voting methods require votes modelling based on rankings and not on continuous values. On the other hand, other methods consider votes with a binary format. Moreover, some voting methods consider a normalised value of votes or similar. Some of the methods in this component use method of the Voting methods operations component.

- Basic operations. This component implements all the basic operations which are used by voting and fitting operations. These operations include sums, rankings, weightings, obtaining maximum, minimum and mean values, transfers vote between configurations, sorting votes, count users, comparisons, and so on. In addition, this component

implements the functionality of the voting methods that require temporal considerations, such as the results of previous votes. To do this, it uses component Voting history record.

- Measure of satisfaction. The set of methods for evaluating voting methods are defined in this component. Different measures of satisfaction are considered, all based on the distance between the winning configuration and the value given to this configuration by a user, as was discussed in Section 3.3. These satisfaction metrics are measured over the services. For this, mean, total, and normalised values are calculated. In addition, the Voting history record can be used to calculate temporal metrics, such as the accumulated satisfaction or the dissatisfaction of users over time.

- Voting history record. This component is used to keep a record of the voting results and of the voters who have participated in the votes and their preferences. In this way, temporal functionality is provided. For this, storage and access functions are provided. Moreover, time is managed as an abstract variable represented by steps, which can be increased with each voting process.

As was commented before, the implemented model considers two different uses. First, using the socepy software to apply the voting methods and obtain the results of the winners and simple satisfaction measures. Second, using the socepy software in a continuous process in which a record of votes and results is registered.

This second case considers the application of weightings and measures that require knowing the information of previous votes in which a voter has participated. For example, control of fair voting can be applied and metrics dependent on previous votes can be calculated, such as the time of dissatisfaction with the service of each user.

In this project, the proposed Social Choice model is used in a blockchain voting platform. Therefore, the information about the votes is stored in the blokchain, providing characteristics such as integrity and security. Thus, the software socepy is used as in the first case commented.

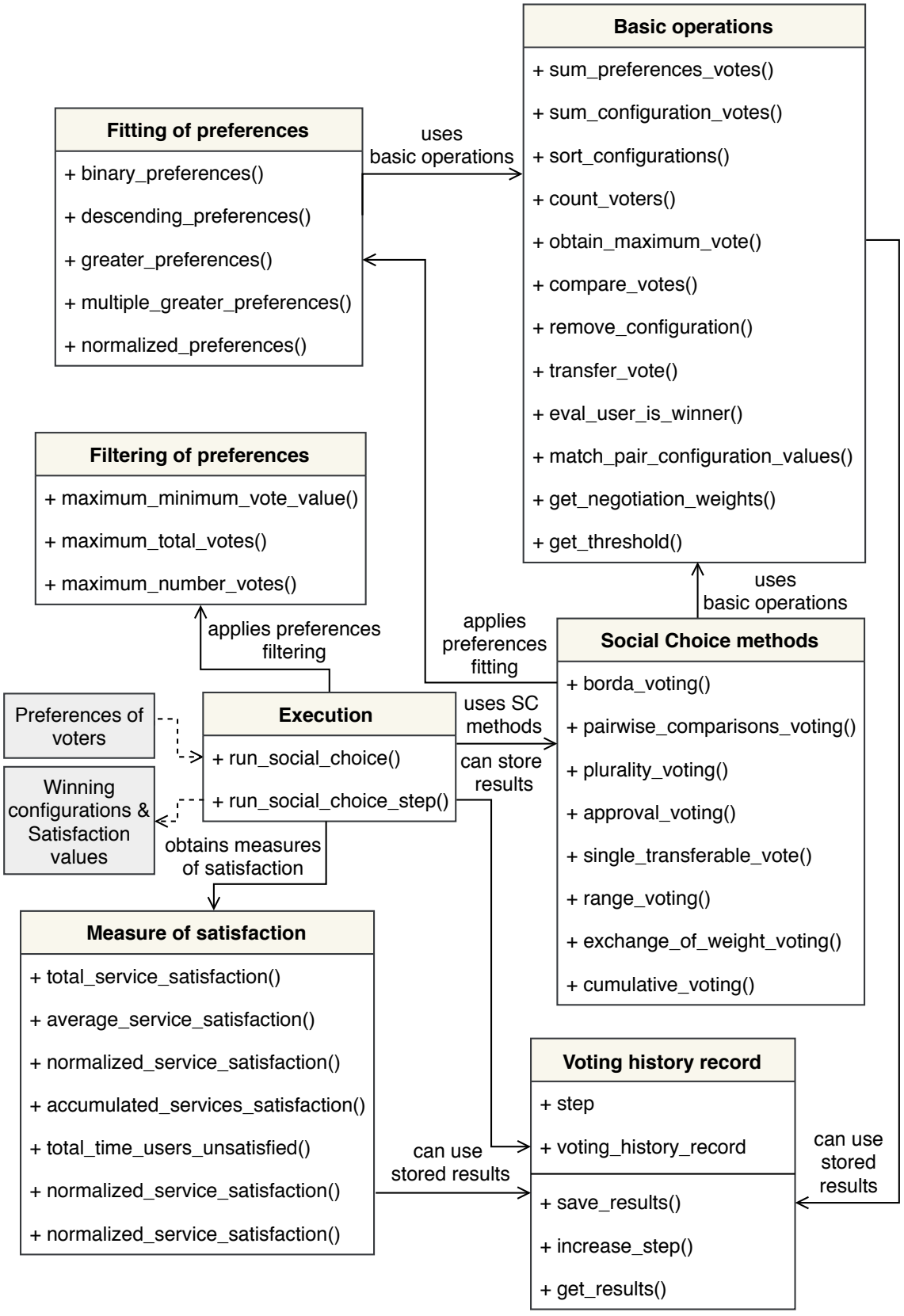

Figure 3.2: Diagram of classes of the implementation of the proposed Social Choice model

CHAPTER 4

Use Case Simulation Model

This chapter describes the simulation model of the case of use proposed in this project. The case of use considers a smart building where occupants can vote the temperature in a room. Thus, this chapter presents the modelling of occupancy, the modelling of HVAC system, evaluation metrics and the scenario considered. In addition, results obtained by the simulation are presented and evaluated.

This section describes the simulation model proposed in this work to study the case of use. It considers occupants in a smart building which configures the temperature in a room applying Social Choice methods. The simulation model proposed to model this scenario is presented in Figure 4.1.

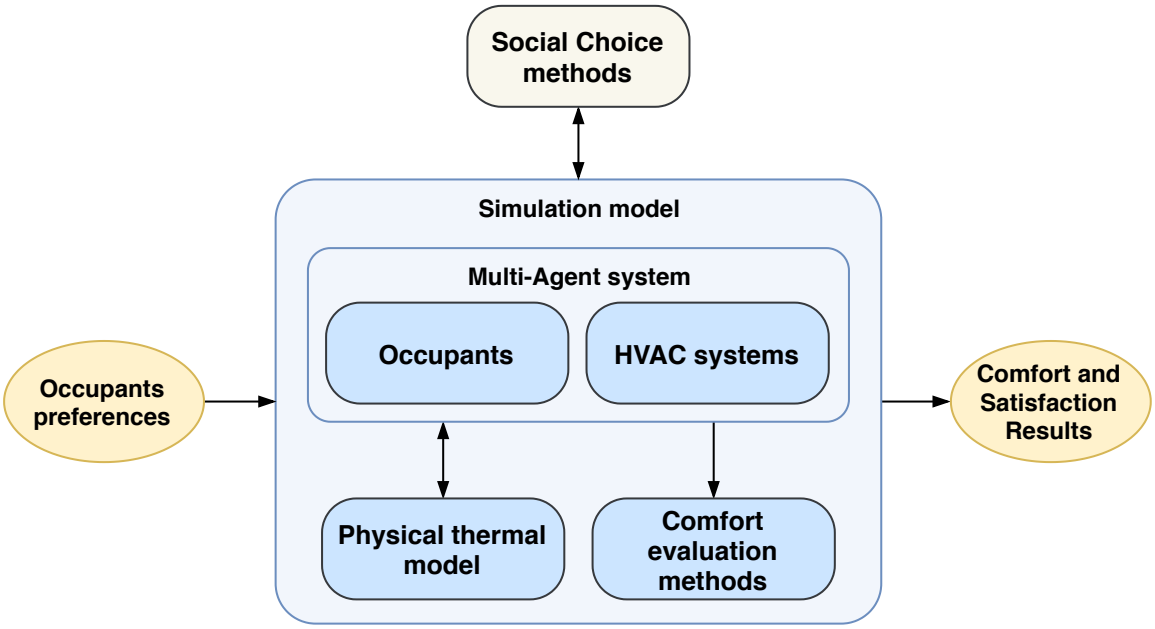

Figure 4.1: Architecture of the simulation model

As can be seen in the figure, the simulation model is organised in four modules, which are (i) the occupants, (ii) the Heating, Ventilation and Air Conditioning (HVAC) systems, (iii) the physical thermal models, and (v) the comfort evaluation methods. Moreover, voting methods from Social Choice theory are considered to maximise the comfort of the occupants. The simulation model uses configuration values about occupants (temperature preferences, activity in the building...) to obtain results of the comfort and the satisfaction of the occupants with different voting methods.

The occupants and the HVAC systems are the two types of agents which form the Multi-Agent system. It is the centre of the simulation model and uses the rest of the modules. Thus, the agents orchestrate the simulation. The behaviour of these agents has been modelled using state machines. The physical thermal models enable to model the temperature. For this, thermal zones and the heat balance method are considered. Comfort evaluation methods are used to know the occupant comfort with the temperature in a room. For this, two methods are considered, which are a standardised Fanger's method and a method proposed based on individual thermal perception.

4.1 Multi-Agent system

The simulation performance is made by the agents. In particular, two types of agents have been considered: the occupants and the HVAC systems. The behaviour of these agents has been modelled using state machines. In this manner, each state represents an agent's action or behaviour.

The occupants in the building are modelled through occupancy agents. These occupancy agents are modelled by various Markov chain states, a schedule, environmental behaviours and preferences of temperature. In addition, the states of the Markov chain represent actions, which have a duration (time of activity) and a position where are performed. In this manner, states are the engine to model the activity of the occupants.

Furthermore, the states of the occupancy agents belong to an inhomogeneous Markov chain. Thus, when a trigger of change of state occurs, the probability associated with each possible new state is different for each time period. A time period is a pair of time instants (t1, t2) with the same Markov matrix. For instance, a possible time period would be the pair (10:00, 13:00). These periods are defined considering the occupants' schedule.

In addition, when a trigger for a change of state occurs depends on both the schedule of the occupant and the time of activity. Moreover, a normal Gaussian random function is applied to the schedules and activity times of an occupancy agent to better represent randomness.

During the simulation, an occupancy agent interacts with the HVAC system exchanging information. In addition, decisions of occupants change according to different environmental behaviours. In particular, the environment behaviours considered modifies the decision to close or not a door that should be closed or let the lighting on. These actions are relevant in the HVAC system consumption. the environment behaviours are different depending on the time of day and the type of occupant. In this model, three different environmental behaviours are considered, which are excellent, good and bad.

Furthermore, the temperature preferences of each occupancy agent are considered. These are required to apply the voting methods. Occupants inform the HVAC system about their temperature preferences to decide the temperature in a room. Of course, they may be different for each occupant.

The HVAC system of the buildings is modelled by HVAC system agents. They are

assigned to the thermal zones, in relation one to one. A thermal zone is a set of rooms with the same air mass and, therefore, the same temperature.

Thus, HVAC system agents control the rooms' temperature evolution during the simulation. In addition, smart sensors enable to obtain information on individual occupancy presence and temperature preferences. Moreover, voting methods to occupancy comfort increase and the comfort evaluation are operated by the HVAC. The performance is regulated by two states: 'on' and 'off'.

4.2 Physical thermal models

Physical models enable the modelling of physical processes, such as the energy exchange. These models enable to simulate the physical phenomena. In particular, a thermal zones model is considered. This regulates the heat and cold gains and losses of the building's thermal zones.

Furthermore, the thermal zone model is performed considering the *heat balance method*. This method is used to model the temperature in a room by applying the first law of thermodynamics, the principle of energy conservation. The application of the heat balance method is done separately in each thermal zone, which might be at different temperatures. Each thermal zone is formed by one or more different rooms sharing the same air mass.

It has been considered the heat and cold transfers generated through lighting, occupancy activity, external walls, inner walls, roofs, windows, air ventilation and air infiltration. Building features and climatological aspects have been defined based on a real building located at the coordinates (O3°42'9.22", N40°24'59.4"). The weather conditions of a month of summer have been considered.

All the technical values employed are obtained from American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE) Fundamentals Handbook [156]. Furthermore, the equations used to apply the heat balance method are presented below [157, 158].

1. Thermal conduction through the roof, walls and windows: $Q = U * A * (T_i - T_o)$, where U is the thermal transmittance of the material, A the element area, T_i the temperature inside and T_o the temperature outside. In the calculation referring to inner walls, T_i and T_o are room temperatures of different thermal zones.

2. In Summer months, the effect of convection, conduction and solar radiation is considered: $Q = U * A * CLT$, where U is the thermal transmittance of the material, A the element area and CLTD the element cooling load temperature difference.
3. Solar load through windows' glass: $Q = A_s * maxSHGC * SC * CLF$, where A_s is the un-shaded area of the windows' glass, $maxSHGC$ the total solar heat transmission, a statistical data, SC the shading coefficient, determined by glazing product effectiveness (∼0.8) and CLF the Cooling Load Factor.
4. Heat gain through lighting system: $Q = (1.2) * (L_{ef}) * W$, where L_{ef} is the light efficiency(∼25%) and W the power (432 Watts).
5. Occupancy loads, which are separated into sensible heat: $Qsensible = N * SHG * CLF$, where N is the number of people in the thermal zone, SHG the Sensible Heat Gain per person (50W) and CLF the Cooling Load Factor(∼0.8); and latent heat: Qlatent=N*LHG , where N is the number of people in the thermal zone and LHG the latent heat gain per person (40W).
6. Thermal exchange through ventilation and infiltration. An outside air entrance for maintaining occupant health and comfort is required: $Q = \frac{ACH*Vol_{air}*\rho_{air}*C_\rho*(T_i-T_o)}{D}$, where ACH is the air changes per hour (4), Vol_{air} the total air volume in thermal zone, ρ_{air} the air density (1.19kg/m^3), C_p the air specific heat (1012J/kg*K), T_i the temperature inside, T_o the temperature outside and $D = 3600$ second/hour. In addition, there is an infiltration, which is a small value and difficult to obtain. Only exchanges by open doors between different thermal zones are considered: $Q = 1.08 * V * \frac{h*w}{2} * \Delta TV = 100 * \frac{\sqrt{h}*\sqrt{\Delta T}}{\sqrt{7}*\sqrt{\Delta T}}$, where h is the door's height, w the door's width and ΔT the temperature difference between rooms.

The sum of all these loads as watts is used to model the temperature in the rooms, by means of the first thermodynamic law $\Delta Tr = \frac{\Delta J}{C_\rho * Vol_{air} * \rho_{air}}$, and specific heat equation $\Delta J = J_{HVAC} - J_{Qload}$, where ΔTr is the temperature increase or decrease in the thermal zone, Vol_{air} the total air volume in the thermal zone, ρ_{air} the air density (1.19kg/m^3), C_p the air specific heat (1012J/kg*K), J_{HVAC} the Joules provided by HVAC system and the J_{Qload} the total Joules obtained from the total thermal load exchange, that is, the result of applying the above equations.

Space where take place the study is formed by rooms. Each room is characterised by width, length, height, number of windows (and their size), a type (office, corridor, restroom ...), inner and external walls, and doors. All the rooms, except the restrooms, belong to a thermal zone. The thermal power of each HVAC system is obtained at the beginning of the simulation. For this, the heat balance method is applied in a worst-case

scenario.

4.3 Comfort evaluation

There are important differences between the desired temperature in a room of one person and another [38]. In fact, this difference also appears in the degree of discomfort of the occupants with a room temperature which is different from personal preference. Therefore, there are complications to measure the occupant thermal comfort in a representative way.

In this work, two methodologies of comfort evaluation have been considered. The first one is known as the Fanger's method [159, 160]. This method is probably the most commonly cited method to evaluate thermal comfort in buildings. According to this, there are six factors which have a higher influence on the thermal exchange between humans and the environment. These factors are the level of activity, the clothing characteristics, the temperature, the relative humidity, the mean radiant temperature and the air velocity. Considering these factors, Fanger's method provides the uncomfortable people percentage in a determined environment.

The evaluation of these factors enables to calculate the value of PPD (Predicted Percentage Of Dissatisfied), which is the percentage of occupants with discomfort with the temperature value. For this, the next equation is applied: $PPD = 100 - 95 * exp(-0.03353 * PMV^4 - 0.2179 * PMV^2)$, where the PMV (predicted mean vote) must be known.

The value of PMV can be obtained using an approximation method, so-called ISO 7730 approximation method [161]. This method is based on tables and is preferably used instead of following the complex analytic method. This approximation method is based on calculating the estimated value of the PVM, which is the PMV0, using a table associated with the set of values considered for the before named factors.

Then, the final PMV value is obtained using the equation $PVM = PVM0 + f_h * (HR - 50) + f_r * (T_r - T_a)$. In this equation, f_h and f_r are the correction of the factor of humidity, 0.008, and the temperature, 0.13; HR is the relative humidity, 0.4; T_a is the environment temperature and T_r is the mean radiant temperature, which is considered one degree more than T_a.

In this work, the values for the Fanger's method factors have been defined considering a Summer month. Thus, these values are an activity level of 120kCal, a clothing level of 0.75, relative humidity of 40%, and an air velocity of 0.15m/s. Considering these values,

the values of temperature and their associated values of PVM0 are the following: 18°C, -1.49; 20°C, -1.00; 22°C, -0.48; 24°C, 0.04; 26°C, 0.56; 28°C, 1.09; 30°C, 1.62; 32°C, 2.17.

According to the above, Fanger's method provides a generic measure that could be obtained knowing the general environment and occupancy characteristics. Thus, this method is really useful when individual occupant preferences are not known. However, Fanger's method does not enable to measure the comfort of an individual occupant. Thus, this method does not consider the occupants' temperature preferences.

According to this, a second analysis method is used in this work, which is called preferences method. This method enables to estimate the occupants' comfort considering the individual preferences. For this, a function is defined. This function evaluates the discrepancy, which is the difference between the room temperature and the occupants' preferences. Thus, a variation in the temperature with respect to the occupant preference causes a rise of the discomfort depending on the value of discrepancy.

The proposed function is a piecewise-defined. The design of this function is inspired on works on the variation in the perception of the thermal comfort, such as as [162, 163]. In particular, this function has three regions, which are a temperature variation of (i) up to two degrees, (ii) between two and four degrees, and (iii) greater than four degrees. Each region has a steeper slope, greater decrement of comfort when there is a greater difference with the desired temperature. Thus, the discomfort variation is greater when there is more distance with the desired temperature.

4.4 Experimentation

The model for improving occupancy comfort proposed in this work is implemented in a simulation tool. This simulation tool considers the use of Social Choice methods which allow occupants to vote the temperature in a room. In this manner, the different voting methods are studied. This section presents the results corresponding to this study.

According to this, this section has two parts. First, the scenario considered to perform the experimentation and the implementation of the tool are described. Second, the results of occupants' satisfaction and comfort obtained by each voting method are presented and evaluated.

4.4.1 Scenario description and implementation

The experimentation has been carried out in Telecommunication Engineering School of the Universidad Politécnica de Madrid. In particular, the floor is shown in Figure 4.2 has been considered. It has a rectangular shape with an area of ∼16x100m2. Moreover, The floor is divided into 37 rooms and 28 thermal zones, distributed as 14 offices, with one thermal zone by office, 20 laboratories, with 12 thermal zones, and 1 hall and 2 corridors, divided into 2 thermal zones. In addition, there is one restroom, which does not belong to any thermal zone, and two exits. Classes are not considered. The maintenance department has been contacted to collect information about the distribution of the thermal zones. There are 1-5 professors in the offices and 3-8 researchers in the laboratories.

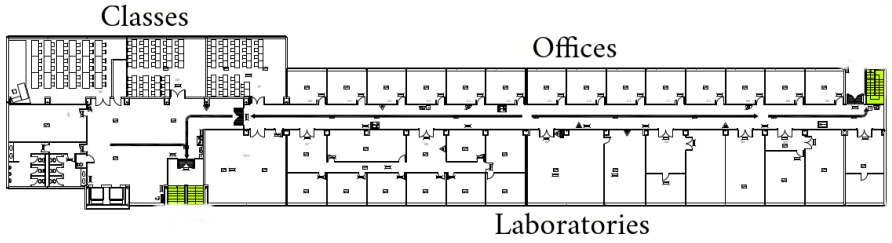

Figure 4.2: Plan of the building considered to carry out the experimentation

Furthermore, a survey has been submitted and processed. This survey was completed by 18 professors and 16 researchers who work in the considered university's floor. In this manner, the information of occupants on the daily activity, environmental behaviour and temperature preferences are known.

First, the information on the daily activity enables to define the place when the states are performed and their duration. Second, the information on environmental behaviour is used to define the percentage of occupants with each behaviour. Finally, the information on temperature preferences is used to define a Normal distribution with an average of 23°C and a standard deviation of 2, which provides an approximate variation of 19.5°C to 26.5°C.

The simulation model proposed is implemented using the programming language Python. Moreover, two packages are used, which are MESA and Transition. Transitions is a lightweight, object-oriented state machine implementation. This package is used to model the behaviour of the agents in the building. However, the Markov state machine behaviour is not considered in this package. Thus, this functionality is implemented.

Mesa [164] is an open-source software useful to create agent-based models. Mesa's architecture is defined with modularity, providing four different modules, which are a Model,

a Scheduler, the Agent class, and Space. This space is defined by means of a grid with coordinates (x, y). The main extensions provided in this work are: (i) definition of agents (occupants and HVAC system), (ii) modelling of crowd's behaviour in buildings, (iii) definition of physical space, which is a discrete space of 0.5x0.5m forming rooms, (iv) modelling of physical models, (v) consideration of comfort evaluation methods and (vi) integration with Social Choice methods. The simulation tool is provided as open-source software in [165].

4.4.2 Social Choice model simulation results

Different voting methods from Social Choice theory which allow occupants to vote the desired temperature in a room are studied. This section presents an evaluation of the considered Social Choice methods, which are described in Section 3.2. For this purpose, the results of comfort and of satisfaction obtained with each method are presented and compared.

Occupants' comfort is measured with the Fanger's method and the proposed preferences method, which are described in Section 4.3. As was discussed, the difference between them is that the preferences method enable to obtain individual information about the occupant comfort of the occupants, considering their preferences, while the Fanger's method considers a standard value of temperature. The satisfaction of the occupants with the voting result is measured using distances, which is described in Section 3.3.

According to this, this section firstly presents and describes graphical results comparing the eight voting methods, and then presents conclusions. The eight voting methods are: (M1) Approval voting, (M2) Borda voting, (M3) Cumulate voting, (M4) Exchange of weight voting, (M5) Pairwise comparisons voting, (M6) Plurality voting, (M7) Range voting and (M8) Single transferable vote. In addition, a situation in which no voting methods are applied is also considered (M0).

Figure 4.3 presents the comfort values obtained by each voting method for a day considering the preferences method. As can be seen, there are three different areas in the graph. The first one and the last one, which are the morning and the afternoon, provide the most valuable information. In contrast, the area corresponding to the lunchtime is a transition area since it is a time of the day of high instability. In this area, the value of the comfort tends usually to a maximum or minimum value because the number of people in a space is greatly and abruptly reduced. Therefore, focusing on the other two areas the next results are obtain. It can be seen how the M0, M1, M3, M8 and M6 methods obtain lower results

than the other methods, which are around or below 75%. The M5, M7, M4 and M2 methods provide comfort values around 80%.

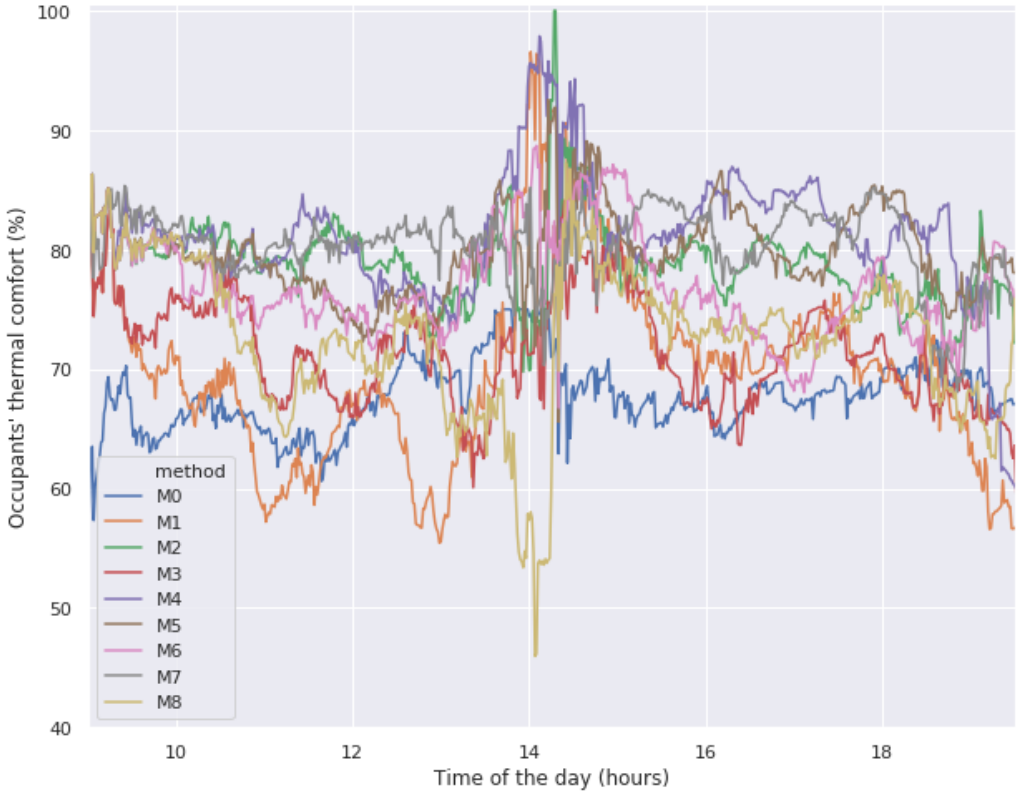

Figure 4.3: Comfort values obtained by each Social Choice method considering preferences method

Figure 4.4 presents the comfort values obtained by each voting method for a day considering fanger's method. As can be seen, the results presented have some similarities to the previous case, considering the method based on preferences. However, there are three main differences.

The first difference is the value of comfort, y-axis. In this case, there is a smaller amplitude range and a smaller variation between the maximum and minimum values obtained by the different voting methods. This is due to the fact that the Fanger's method considers a fixed value of intermediate temperature, being more stable than considering the personal values of the occupants, which can be more distant values.

The second difference corresponds to the results. There is a modification in the position of some voting methods if they are ordered from better to worse. For example, the M8 method presents, in relation to the rest of the methods, a better result than with the method of preferences.

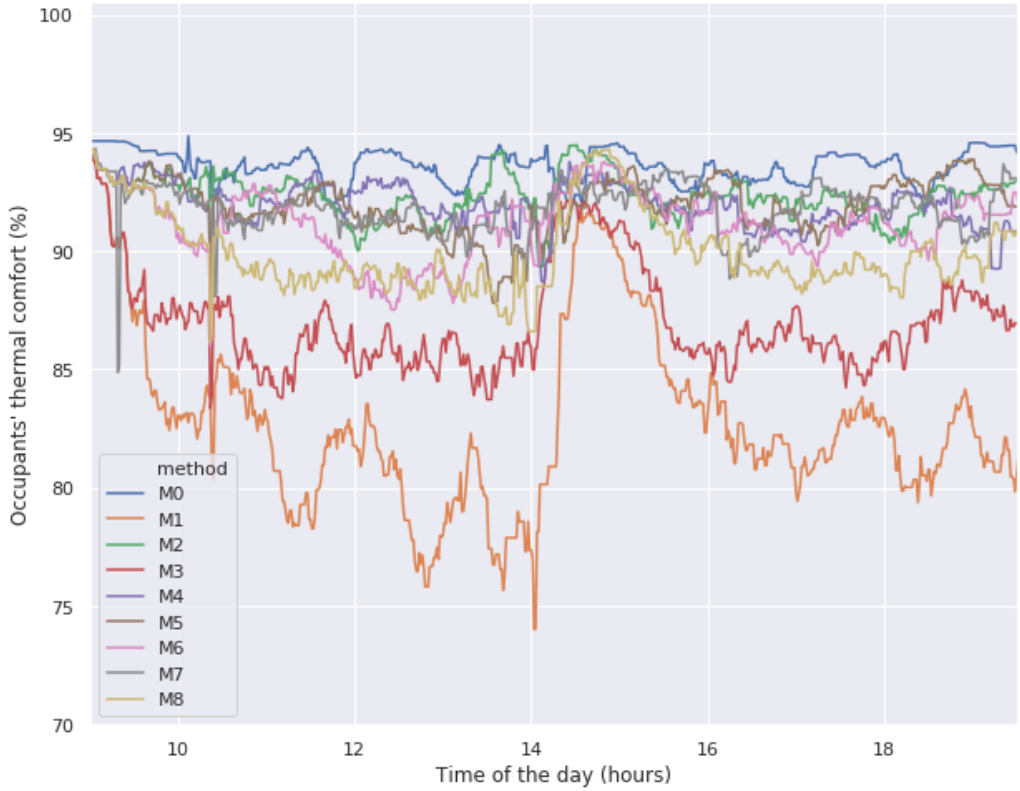

Figure 4.4: Comfort values of each Social Choice method considering Fanger's method

However, similar results are obtained in general with both comfort evaluation methods. In particular, the M1 and M3 methods are inferior to the rest, with values around 83% and 87%. The M8 and M6 methods get intermediate values, which are around 90%. The M2, M4, M5, M7 methods are above the others, with values around 92%.

The last difference refers to M0, when no voting method is considered. In this case, a continuous value of 24ºC of temperature is considered. This value is the optimum of Fanger's method. Thus, the result obtained with the Fanger method corresponding to M0 can be seen as an upper limit, which is not perfect because there are temporary variations of the temperature value due to the thermal model.

That is, therefore, the expected result since the Fanger's method does not consider the individual preferences of the occupants, which is the objective of the M1-M8 methods. Therefore, considering an alternative method to evaluate individual comfort as the proposed preferences method is interesting.

Moreover, taking into account this fact about the result obtained with the optimal case considering Fanger's method (M0), the following information can be stated. The voting

methods that obtain a better result considering an evaluation with the Fanger's method, are those methods that obtain a better collective preference combinating the individual preferences.

This is because the Fanger's method considers a mean, generic, 'suitable for everyone' value. Therefore, the voting method which obtains a value closer to the value of M0 case is the voting method that adequately considers all voting participants. This information may be false since the methods that have a better result with Fanger's method could not consider individual preferences.

However, some veracity can be affirmed because of the results obtained with the preferences method are similar. That is, the voting methods that obtain the best results with Fanger's method also obtain the best results with preferences method. Moreover, these voting methods do not achieve values closer to the maximum with the Fanger method because the average temperature preference of occupants is not 24°C but 23°C. As was commented above, this value is defined using the information given in the survey.

Figure 4.5 presents the results of the average occupant satisfaction provided by each voting method. In this case, methods M1 and M3 are clearly below the rest with satisfaction values below than 5. The M8 and M6 methods provide a medium satisfaction value, which is between 6 and 7. The methods that stand out as the best are M2, M5, M4 and M7, with satisfaction values between 7 and 8.

Moreover, the M0 case obtains very low results of satisfaction. This could be expected since in this case there is no adaptation of the temperature. However, these results are not so low when are compared with the two voting methods that obtain the worst results. This may mean that sometimes the application of an inappropriate voting method is no better than not considering one voting method.

Moreover, Table 4.1 presents a comparative summary of the Social Choice methods studied. Each method is presented with its value of comfort and average satisfaction obtained. As it can be seen, and according to the above, the voting methods can be divided in three groups: (i) Range voting (M7), Exchange of weight voting (M4), Pairwise comparisons voting (M5) and Borda voting (M2), which obtain the best results; (ii) Plurality voting (M6) and Single transferable vote (M8), which provide medium results; and (iii) Cumulative voting (M3) and Approval voting (M1), which obtain the worst results. This classification is made considering the average satisfaction metric. However, there is a strong relationship between the values of satisfaction and comfort obtained by each voting method.

Figure 4.5: Average satisfaction of each Social Choice method

According to the above, the Social Choice methods obtaining the best results are generally those that consider grade information about the preferences. In particular, Range voting and Exchange of weight voting methods consider the sum of the preference scores between -10 and 10. Between these two methods, Exchange of weight voting obtains a slightly worse result since this method is defined to favour minorities.

Borda voting and Pairwise comparisons voting methods also provide good results. This is because they also consider a rating of the voting alternatives. However, the difference with Range voting is that they consider fixed ratings (decreasing steps), providing less information. Again, Pairwise comparisons voting method favors the minorities, providing a slightly worse result.

Plurality voting and Single transferable vote methods obtain intermediate results. This is because these methods only reflect a single option as a preference, providing less information. In addition, the Single transferable vote method obtains worse result due to the consideration of the minorities.

The worst results are obtained by Cumulative voting and Approval voting methods.

This is because Cumulative voting limits the total number of votes that can be distributed to the options. Thus, this method limits the information and generates noise. In addition, Approval voting method considers multiple options as equally valid regardless of their grade. Because of this, it provides more noise since one option can not be defined as preferred.

Method	*Id*	*Satisfaction*	*Comfort*	
			Preferences method	*Fanger's method*
Range voting	M7	7.67	79.45	91.68
Exchange of weight voting	M4	7.49	80.58	91.86
Pairwise comparisons voting	M5	7.19	78.79	91.91
Borda voting	M2	7.18	79.08	91.93
Plurality voting	M6	6.44	76.21	90.42
Single transferable vote	M8	6.18	74.84	90.19
Cumulative voting	M3	3.74	71.66	87.01
Approval voting	M1	3.58	69.12	82.98

Table 4.1: Average satisfaction and comfort of each Social Choice method

Two facts are obtained from this information. The first one is the importance of allowing voters to grade voting options. In this manner, voters not only define a preferred option but also can give information about all preferences. The second one is related to this, the voting methods that obtain the best results are those that consider more information about the preferences of voters. Due to this, it is important a good fit in the modelling of votes according to the voting method chosen.

Moreover, if that fact is not taken into account, valuable information can be wasted or noise can be generated. However, this facts are affirmed in this case of study. As has been commented in the analysis of the problem, it is important to study each particular case to discover the most appropriate conditions for applying Social Choice theory.

CHAPTER 5

Blockchain voting platform

This chapter presents the voting platform developed in this work. For this, the architecture of the solution and its components are described. In addition, the component which implements the functionality of voting is described in detail. Finally, the ontology proposed in this work is presented.

The voting platform which enables to apply Social Choice methods in a smart environment is implemented considering the use of Hyperledger Sawtooth project. Section 2.4.4 presents more information about this project and the reason it is considered in this work. This section presents the architecture of the voting platform and describes its main components.

In addition, smart contracts are those that provide blockchain functionality. In particular, Sawtooth considers transaction families. Therefore, this section also describes the transaction family implemented to provide voting functionality. Moreover, this section presents the definition of an ontology considered in this work for use in a REST interface of the voting platform.

5.1 Architecture of the platform

Figure 5.1 presents the architecture of the voting platform based on Sawtooth and its components. As can be seen, the platform is formed by multiple validation nodes connected to each other forming a network. Each of these nodes is identical, that is, they are formed by the same components. In addition, if they are working correctly, they eventually have the same exact copy of the Blockchain as the rest of the nodes.

Therefore, it is not necessary to have more than one node for the platform to work. Moreover, it is the situation that is considered during the development phase. In the final deployment, the appropriate scenario is that each validating node is in as many devices as possible, such as sensors and smartphones. However, there can also be several devices sharing the same node, such as sensors in the same room.

According to above, Figure 5.1 presents a node validator and the components that form each validator node. These components are services running on Dockers containers communicated by TCP connections. Docker containers are lightweight virtual machines which contain all of required for the application operation, such as libraries and other dependencies. A scenario of Dockers containers is defined in a configuration file, which is presented in Appendix C.

In particular, there are six components in a validator node, which are the following: (i) an API REST server (ii) a shell client by command line, (iii) a validation service, (iv) a consensus engine, (v) a configuration service, and (vi) the transaction family with the Social Choice application. All these components are described below.

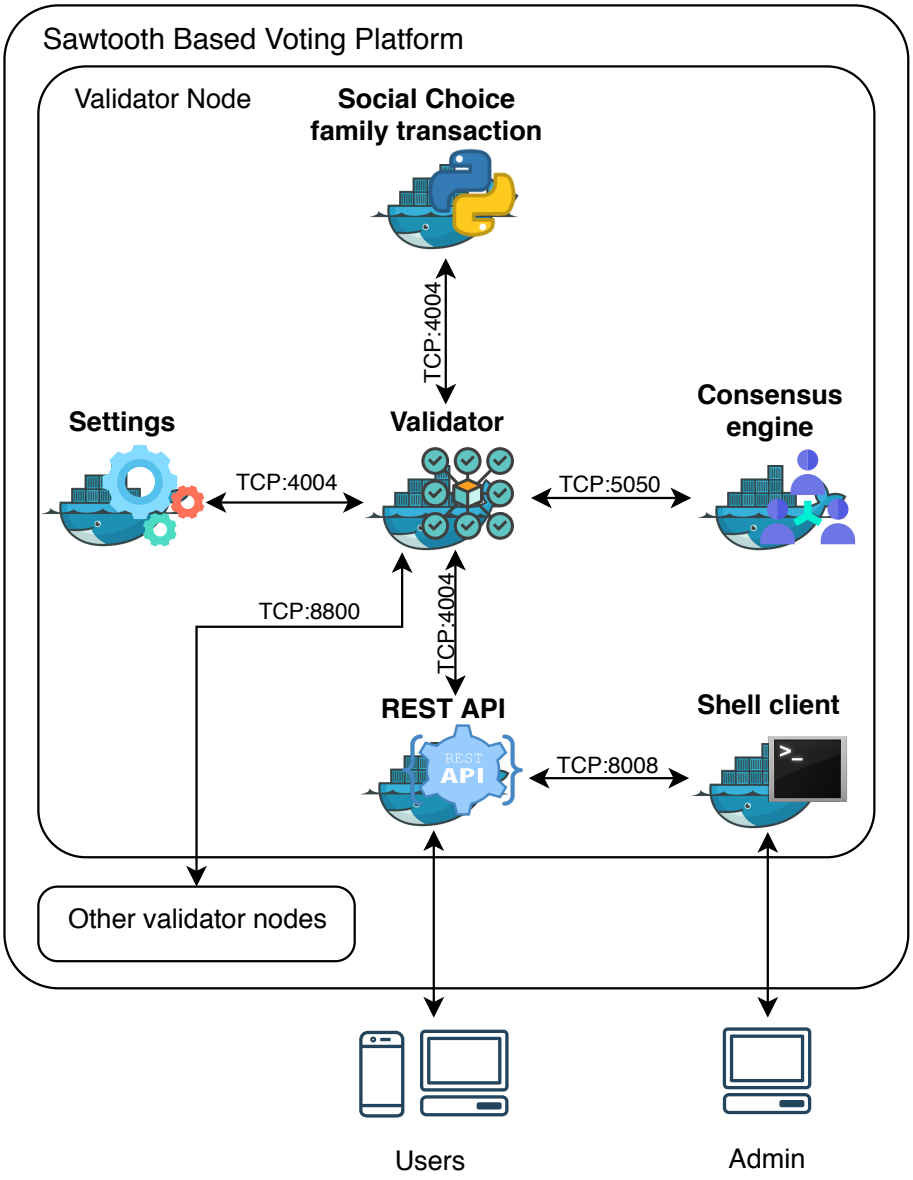

Figure 5.1: Architecture of the voting platform based on Sawtooth

The validator is the core component for running the Sawtooth Distributed Ledger. It is the responsible for validating batches of transactions, combining them into blocks, maintaining consensus, applying the configuration, and coordinating communication among the clients, other validator nodes of the network, and transaction families.

Moreover, the validator starts the voting platform. For this, it initialises the distributed ledger generating the genesis block with the initial configurations, and initialises and interconnects each component of the validator node. In addition, the validator can be controlled using the REST API.

The consensus engine or algorithm is the engine to building agreement among a group of validator nodes in the network. This project is developed and published in the repositories using the Docker image of Sawtooth's 'dev mode engine'. This is a consensus engine for Sawtooth that offers a simplified random-leader algorithm.

This version of the consensus engine is useful for testing applications built on top of the Sawtooth system. Thus, this is recommended to develop transaction families. This is because it has a high commit rate, providing quick feedback to the developer. Although this consensus engine can be used in final development, it has a very inefficient fork-resolution algorithm and makes no guarantees about crash fault tolerance.

Therefore this consensus engine should be changed in a production environment. This operation is very simple, it just requires to change the image that is used in the deployment of the system. For example, in order to use the consensus algorithm with PoET, the image used is changed to 'sawtooth-poet-engine'.

The settings component provides a method for storing on-blockchain configuration settings. The settings stored in the state by this component play a critical role in the operation of the validator node. For example, the consensus module and the pluggable components such as transaction family implementations use the settings during their execution. The setting data consists of key/value pairs.

The Social Choice transaction family is the transaction family implemented in Python which enables to provide the Social Choice functionality. For this, it validates transactions and updates state based on defined rules. This component defines the data model and the business logic for apply Social Choice methods on a blockchain by submitting transactions for different actions. This component is described in detail in Section 5.2.

The REST API component provides a service which enables to interact with the validator using HTTP/JSON standards. In this way, transactions, which produces actions of a transaction family, can be executed. In addition, the API endpoints include RESTful references to resources stored in the Sawtooth ledger which are of interest to users, such as blocks and transactions.

As was commented in section 2.4.4, the REST API process runs as a separate process from the validator process. It treats the validator as a black box, simply submitting transactions and fetching the results. Therefore, a proxy server must be used to consider authentication in operations to query instances of the blockchain and to generate transactions, which can be managed by transaction families. The development of this proxy is

proposed as future work.

The REST API uses a JSON envelope to send metadata back to clients in a way that is simple to parse and easily customised. Moreover, Sawtooth format the data in the payload as comma separated values. Section 5.3 presents a future alternative to change the format of the data provided by the API REST to JSON-LD. An example of the use of this rest service is presented in Section 6, considering a mobile application as a client.

The shell client is the component used to run commands that act as a client application. For this, they communicate with the validator through the REST API. Figure 5.2 presents the elements that form the shell client, which are the Command Line Interface (CLI), the Social Choice commands and the REST Client.

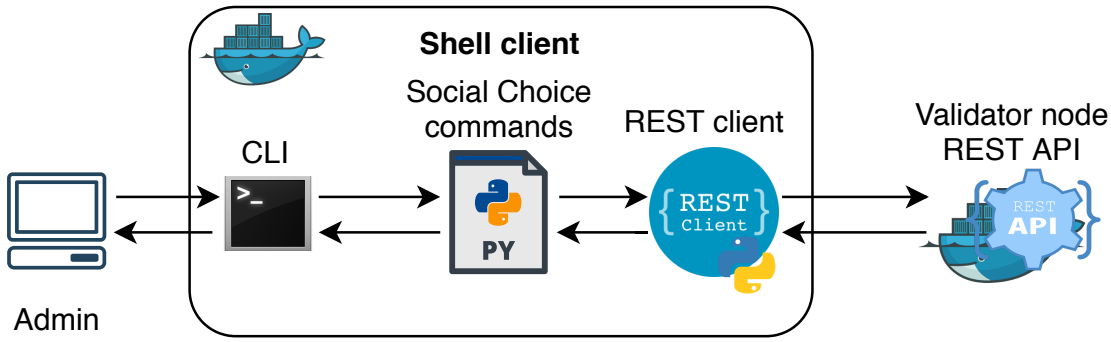

Figure 5.2: Components of the shell client

The CLI is modified to provides a new set of commands. A user (usually admin or developer) run the commands in the CLI to use the services of the Social Choice transaction family. The Social Choice commands are implemented in a Python file that provides the functionality and non-functional requirements, such as the parsing of arguments and the help option.

In particular, eight commands are considered. They have different options and arguments, which can be obligatory (voter and voting information) and optional (authorization, REST service endpoint URL, private key...). These commands enable to perform the following actions: (i) create a voter, (ii) create a voting, (iii) register a voter in a voting, (iv) unregister a voter from a voting, (v) apply the voting method of a voting, (vi) define voter's preferences for a voting, (vii) obtain information from an entity (voter or voting), and (viii) obtain information from all defined entities. Section 5.2 presents the implementation of these actions.

The commands execute functions on a REST client. It is implemented to make requests with the appropriate format and considering the arguments given by the user in the terminal.

The REST client communicates with the REST server of the Validator node to obtain information of the state of the blockchain and to execute transactions. The state and the transactions are the ones corresponding to the Social Choice transaction family.

5.2 Social Choice transaction family

In blockchain technologies, smart contracts are programs that take a transaction as an input, process it, and produce an output. Smart contract provides the business logic, implementing the functionality for a specific case of use. In Hyperdleger Sawtooth, the concept of smart contracts is expanded by viewing a smart contract as a 'transaction family'.

The validators pass the transactions through the distributed log and route them to an appropriate transaction family. These transaction families ingest the payload of the transaction and processing it, generating a new distributed state in the blockchain. This state is managed by storing a serialized state of in a Radix Merkle tree. For this, each transaction family is assigned to a namespace to which it can write.

A Social Choice transaction family is implemented using the Sawtooth Python SDK. It enables to develop a Social Choice application over Sawtooth. Figure 5.3 presents the four components of this transaction family, which are the following: (i) the transaction handler (ii) the payload checker, (iii) the Social Choice methods, and (iv) the Social Choice state. In addition, both the transaction handler and the Social Choice state communicate with the validator of the validation node to which that social choice transaction family belongs. All four components are described below.

The transaction handler is the core component of the transaction family. It defines the business logic and is connected with the validator and with the rest of the components of the Social Choice transaction family. Mainly, the transaction handler has four responsibilities. First, it is responsible for registering the transaction family in the validator, enabling that it can be located and used.

Second, the transaction handler manages the transaction payloads and associated metadata. Third, it filters and manages the actions defined in the transactions to use the application. Finally, it applies the methods of getting and setting state when actions require it.

The routing of transactions to a transaction handler is made by the validator. For this, the transaction handler is addressed using identification properties. They determine the set

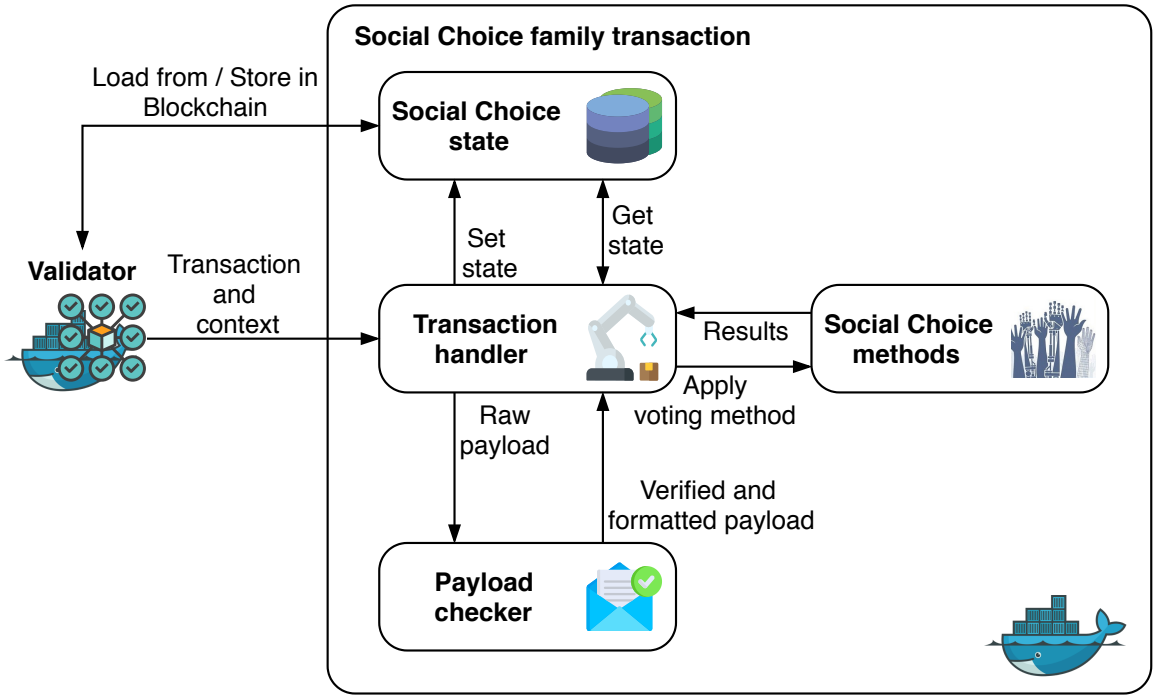

Figure 5.3: Architecture of the transaction family of Social Choice application

of transactions of a transaction family that can be processed by the transaction handler. In particular, there are three identification properties.

These are the family name, which is the name of the transaction family; the family version, which is a list of versions of the transaction family, so different versions of the same transaction family can be supported; and the namespace, which is a list containing all the handler transaction's namespace. This namespace is the same as those used to address the data stored in the Merkle-Radix tree in the state managing.

The transaction handler is responsible for filtering and applying six different actions. These are defined in the payload of the transactions. The requests which enable to access to data loaded in the state of the blockchain are not the responsibility of the handler but of the validator. This is because the access to data does not produce changes of state and therefore they are not originated by transactions. In particular, six actions are implemented, which are described below.

- Create a voter. This action enables to create a new voter and stores it in the blockchain. This action has two obligatory parameters, which are the voter id or

username and the public signature. In addition, it has one optional, which is the preferences, which is initialized as empty if not is given.

- Create voting. This action creates new voting and stores it in the blockchain. For it, three obligatory parameters are considered, which are the id of the voting, the possible configurations or candidates and the method of Social Choice theory.

- Register a voter. This action enables to load a voter and a voting already created and stored in the blockchain using their ids and modifies their argument 'preferences'. In this manner, a voter with its preferences can be included in the preferences of a voting. Moreover, the voting is included into the preferences of a voter, if this was not done previously. In this way, voting considers the voter when it applies Social Choice methods. If a voter had not defined preferences for the voting, these are initialised to zero. Once this is done, both the voting and the voter with their new preference argument are stored in the blockchain. This action requires two obligatory parameters, which are the voter's id or username and the voting name.

- Unregister a voter. This action gets a voting already defined and stored in the blockchain to modify its parameter 'preferences'. Afterwards, voting with the attribute modified is stored again in the blockchain. In this way, a voter is removed from the preferences of a voting. Therefore, voting does not consider the voter when applying Social Choice methods. The voter's preferences are not modified so that they can be used in a new register. This action requires two obligatory parameters, which are the voter's id or username and the voting name.

- Set preferences. This action enables to get a voting and a voter already defined and stored in the blockchain to modify their parameter 'preferences'. Afterwards, the new voting and the new voter are stored again in the blockchain. In this way, a voter's preferences corresponding to a voting are modified. This action requires three obligatory parameters, which are the voter's id or username, the voting name and the new preferences.

The transaction handler applies these methods by interacting with the other components of the family transaction and with the validator. The transaction handler receives from the validator a state context and each transaction with the data (action and parameters) and the metadata.

The context provides the interface for load data from, remove data from and store data in the state. It is given to the Social Choice state component enabling it can interact with

the validator and the blockchain. In this manner, the transaction handler can use the Social Choice state component to getting data from, setting data in, and deleting data from the state.

The Social Choice state defines a data model. It is formed by two instances, which are voter and voting. Figure 5.4 presents the model of data, which is formed by the instances and their attributes. As can be seen, voter has a public sign, a id or username and a dictionary of preferences; and voting has a name, a list of configurations, a list of winners, a of Social Choice method and a dictionary of preferences. Both type of instances are related by the preferences attribute, which is a dictionary with the value of preference of a voter to each configuration of a voting.

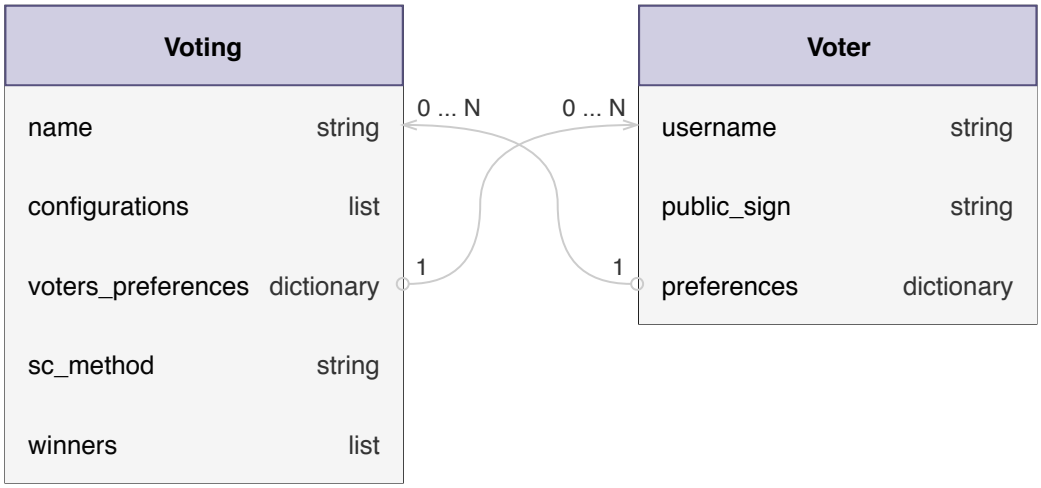

Figure 5.4: Model of data of the voting application

Therefore, the Social Choice state component defines the data model of the state and implements the functions to manage the data in the state. These functions includes the operations get (load), set (store) and delete, which are applied considering the hash of the namespace and of each instance.

In addition, serielization and deserialization functions are defined. These functions enable to transform values and objects which are non-codeable, such as integers and dictionaries, to strings and vice verse. This is beacuse the data must be coded to be loaded in the state.

The transaction handler use the payload checker component to manage the payload. This component works as a set of filters which decode the payload of the transactions and check that the data is correct, releasing an appropriate exceptions if they are not. In addition, they apply formatting operations. This is because the parameters are given in the

payload as strings. Therefore, conversion operations must be applied to integers, lists, and dictionaries.

The Social Choice methods component is used by the transaction handler to apply voting methods. For this, this component requires the value of the attribute of a voting with the preferences. In this manner, the results of the voting winners are obtained and the voting is updated. The model and implementation of this component is described in Section 3.

In addittion to this, although a client is not presented in Figure 5.3, it is theoretically considered part of the family transaction. This is because it must use the same data model, serialization/encoding method, and addressing scheme. The client is responsible for managing the client logic of the application, creating and signing transactions, combining those transactions into batches, and submitting them to the validator. According to this, a based on smart phone application client is presented and described in Section 6.

5.3 Proposed ontology

The definition of an ontology enables to move the information of the systems from data to semantic concepts. This increases the precision and efficiency in the use of information. Moreover, ontologies promote greater consistency and understanding of the meaning of information to humans and computing systems. In fact, an ontology is useful to integrate different systems.

In this work, an ontology is defined. It describes semantic concepts, relations and properties used in the voting platform. Thus, this ontology considers the aspects concerning to the Social Choice model. In particular, the proposed ontology is used to define a knowledge exchange system based on the REST API provided by the voting platform.

One of the phases of the methodology considered to build an ontology, which is described in Section 2.5.1, is the consideration of already defined ontologies. At the time of doing this work, there are some ontologies that define voting terms and relations. However, these ontologies are related to political elections and not to Social Choice theory. Therefore, they are not used in this work.

Instead, the use of ontologies that define particular concepts used in the proposed Social Choice model, such as voting and voter, are considered. In particular, the DBpedia is used, which is a project for the extraction of data from Wikipedia and its conversion to a semantic Web version.

In addition to this, four ontology languages are used. These RDF, RDFS, XSD and OWL, which are described as follows. First, RDF provides the definition of types of domains or subject. Second, RDFS enables to describe the domains and the ranges in relations and the comments about the defined classes. Third, XSD allows ontology to define primitive types, such as list, integer and double. Finally, OWL enables to define the classes, the object properties and the data properties.

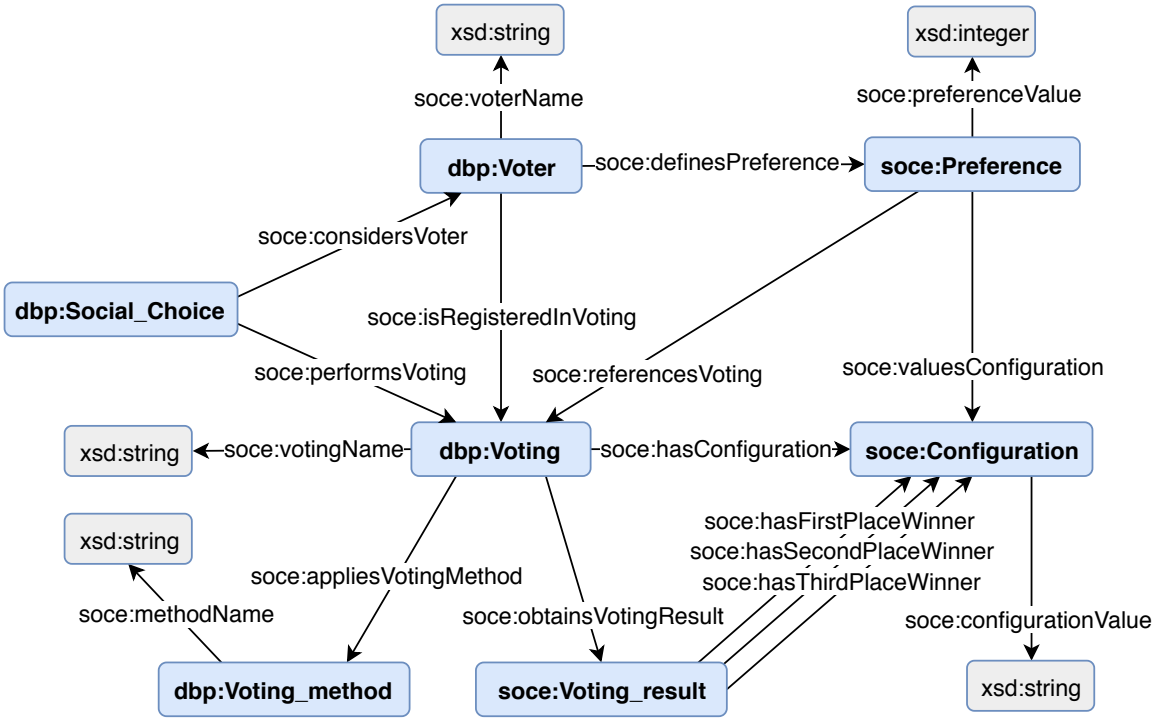

Figure 5.5: Ontology proposed to define the concepts of the voting platform

Figure 5.5 presents the complete graph of classes, relations and properties of the proposed ontology. As can be seen, the ontology is formed by seven classes. The identifier of the class has the form 'prefix:name'. Two prefixed are used, which are 'dbp' for the classes defined in the DBpedia , and 'soce' for the classes defined in this work. Table 5.1 present the list of the classes with their description.

Moreover, twelve object properties are described in the ontology. These properties working as relations between classes. For example, 'definesPreference' represents the relation between voters and their preferences for each configuration in voting. All the object properties are presented in Table 5.2. In addition, this table shows the domain and the range of the object properties. They represent the classes between which there are relation and its direction.

In addition, the ontology defines five data properties, which are presented in Table 5.3.

Class / Term	*Ontology (Prefix)*	*Descripton*
Social_Choice	DBpedia (dbp)	Represents an Social Choice procedure
Voter	DBpedia (dbp)	Represents a participant of votings in a Social Choice procedure
Voting	DBpedia (dbp)	Symbolize a voting process in a Social Choice procedure
Voting_method	DBpedia (dbp)	Represents a voting method of a voting
Preference	Proposed (soce)	Represents a voter's preferences for a configuration of a voting
Configuration	Proposed (soce)	Symbolize a candidate of a voting
Voting_result	Proposed (soce)	Represent a result obtained applying a voting

Table 5.1: List of classes of the proposed ontology and their description

Object property	*Domain*	*Range*	*Descripton*
considersVoter	dbp:Social_Choice	dbp:Voter	Relates an Social Choice procedure to a set of voters
performsVoting	dbp:Social_Choice	dbp:Voting	Relates an Social Choice procedure to a set of votings
isRegisteredInVoting	dbp:Voter	dbp:Voting	Describes which voters participate in which votings
definesPreference	dbp:Voter	soce:Preference	Represents the preferences associated with a voter
valuesConfiguration	soce:Preference	soce:Configuration	Relates a scoring to a configuration
referencesVoting	soce:Preference	dbp:Voting	Relates a preference to a voting
appliesVotingMethod	dbp:Voting	soce:Voting_method	Describes which voting method applies a voting
obtainsVotingResult	dbp:Voting	soce:Voting_result	Presents the results obtained by a voting
hasConfiguration	dbp:Voting	soce:Configuration	Describes the possible settings that a voting has
hasFirstPlaceWiner	soce:Voting_result	soce:Configuration	Presents the first place winner of a voting
hasSecondPlaceWiner	soce:Voting_result	soce:Configuration	Presents the second place winner of a voting
hasThirdPlaceWiner	soce:Voting_result	soce:Configuration	Presents the third place winner of a voting

Table 5.2: List of object properties (relations) of the proposed ontology and their description

These properties represent attributes given to some classes. For this, the classes are defined as domains and the types of value of the attributes as ranges. For instance, the class 'Voter' has a string identifier 'voterName'.

The ontology proposed is evaluated using SPARQL and Jena Fuseki. These technologies have been presented in Section 2.5.2. In summary, they provide a way to create, update and access graphs. Thus, a set of examples are generated to know how the API REST works

Data property	*Domain*	*Range*	*Descripton*
voterName	dbp:Voter	xsd:string	Describes the username of a voter
preferenceValue	soce:Preference	xsd:integer	Describes the score of a preference for a configuration
configurationValue	soce:Configuration	xsd:string	Represents the value of a possible configuration
votingName	dbp:Voting	xsd:string	Represents the name of a voting
methodName	soce:Voting_method	xsd:string	Represents the name of a voting mehtod

Table 5.3: List of data properties (properties) of the proposed ontology and their description

with the ontology.

As also was discussed in Section 2.5.2, the formats considered to specify the ontology are Turtle and JSON-LD. Because of the goal of this ontology is the definition of information provided by a REST API, the format used is JSON-LD, which is compatible with JSON processing. The proposed ontology and an example of use are presented in Appendix D. For instance, using JSON-LD format, the data response given by the REST API is transformed from Listing 5.1 to Listing 5.2.

```
'VoterUsername', 'VoterUsername', '{"Temperature": {"22": -6, "24": 5, "26": 8},
...}'

'Temperature', "['22', '24', '26']", 'borda-voting', '["22", "24", "26"]', '{"
VoterUsername": {"22": 8, "24": 5, "26": -6}}'
```

Listing 5.1: REST service response

```
"@graph": [
{
"@id": "soce:Voter_1",
  "@type": ["owl:NamedIndividual", "dbp:Voter"],
  "soce:definesPreference":[{"@id":"soce:Preference_Temperature_1"}, ...],
  "soce:isRegisteredInVoting": [{"@id": "soce:Voting_Temperature"}, ...],
  "soce:voterName": "VoterUsername"
},
{
  "@id": "soce:Preference_Temperature_1",
  "@type": ["owl:NamedIndividual", "soce:Preference"],
  "soce:preferenceValue": 8,
  "soce:referencesVoting": {"@id": "soce:Voting_Temperature"},
```

```
    "soce:valuesConfiguration": {"@id": "soce:Configuration_Temperature_1"}
  },
  {
    "@id": "soce:Configuration_Temperature_1",
    "@type": ["soce:Configuration","owl:NamedIndividual"],
    "soce:configurationValue": 22
  },
  {
    "@id": "soce:Voting_Temperature",
    "@type": ["owl:NamedIndividual","dbp:Voting"],
    "soce:appliesVotingMethod": {"@id": "soce:Voting_Method_Borda"},
    "soce:hasConfiguration": [{"@id": "soce:Configuration_Temperature_1"}, ...],
    "soce:obtainsVotingResult": {"@id": "soce:Voting_Result_1"},
    "soce:votingName": "Temperature"
  },
  {
    "@id": "soce:Voting_Method_Borda",
    "@type": ["owl:NamedIndividual","dbp:Voting_method"],
    "soce:methodName": "borda-voting"
  },
  {
    "@id": "soce:Voting_Result_1",
    "@type": ["soce:Voting_result","owl:NamedIndividual"],
    "soce:hasFirstPlaceWinner": {"@id": "soce:Configuration_Temperature_1"},
    ...
  },
  ...
  ]
```

Listing 5.2: REST service response using the proposed ontology

CHAPTER 6

Mobile client application

This chapter describes the mobile application implemented in this work. This application is proposed as a solution that can be used as a client of the voting platform. Therefore, this chapter first describes the design and implementation of the application. Then, it presents aspects related to the modules required for its functioning. Finally, this chapter presents the functionality of the application and a description of a possible deployment.

Smart cities and smart buildings consider smart environments which can obtain information from people's smartphones. For this, different ways of connection such as IP connection over cellular data-link service are used. In this work, a mobile application that allows users to interact with the voting platform is designed and implemented.

This application must work on the Android and iOS operating systems. For this reason, React Native framework is used. As was discussed in Section 2.6, It is a solution to cross-compiled application development which provides plenty of advantages over traditional means of mobile phone development.

6.1 Application development

Before starting with the implementation, the functional requirements of the application are defined. First, users can know the votes in which they are registered. Second, users can know relevant information about a vote: name of the vote, current winner of the vote, and voting method applied. Third, users can define new preferences and know them and modify them when preferences have already been defined. Fourth, all the information presented to users can be updated and show the latest situation. Finally, users can modify their username and password.

Once the functional requirements are defined, preliminary design or mock-up can be made. React Native uses components to define the views. Therefore, the design considers the components that will be necessary, their hierarchy and interrelation. Figure 6.1 presents a mock-up of the application considering the required components.

All the components of React Native which have been used in the implementation of the mobile phone application are presented below. In addition, their operation and the purpose for which they have been considered is described.

- View. A component which works as container and is the most fundamental for building a UI. View enables to map directly to the native view. In this application, it is used to organise the definition of the components. This is interesting to apply the same style to a group of components, creating relations of layout among them.

- HomeScreen. This component is implemented to include all the elements used in the initial view. It shows the voting list and the buttons to go to the settings view and to refresh the list. In addition, it incorporates all the required functionality, such as

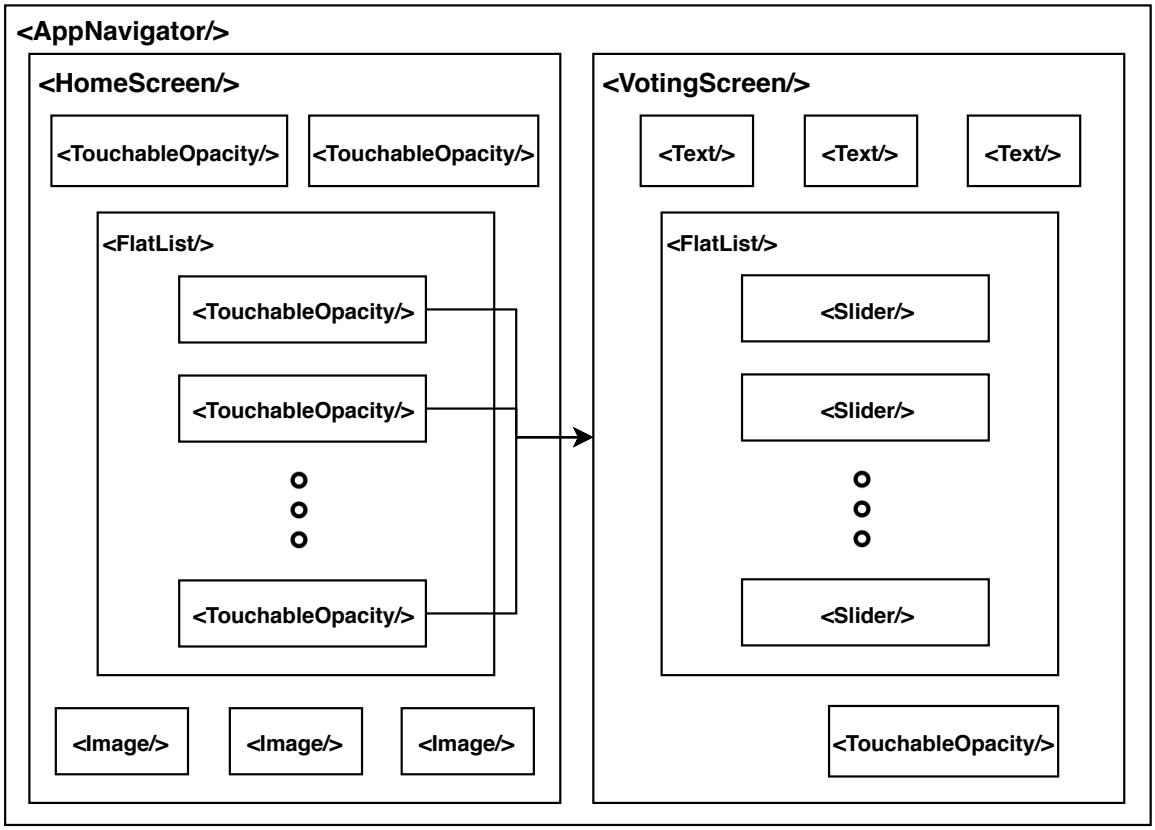

Figure 6.1: Mock-up of the React Native application

obtaining the voting list from the voting platform.

- VotingScreen. This component renders the elements of the voting view. It includes the information fields, the sliders of the configuration and the button to send new preferences. In addition, it implements the functionality for obtaining and sending information about the votes and configurations from the voting platform.

- SettingsScreen. Component corresponding to the user settings screen, which shows the username and password. In addition, it includes the button for changing parameters and its functionality.

- AppNavigator. This component use React navigation methods to manage the presentation of multiple screens and the transition between them. In particular, it is used to navigate between the three screens: home, voting and profile. In addition, this component provides functions to pass attributes between views. For instance, the voting name selected from the voting list is passed to the voting view.

- Flatlist. A performant interface for rendering simple flat lists. Using this component, two lists of elements are implemented. First, a list with the votes in which a user is

registered. Second, a list of configurations of each voting, which can be used by the users to define preferences.

- Image. A component for displaying different types of images. This component has been used twice. First, on each touchable element to represent its functionality. For example, one touchable element contains a 'reload' image to inform that it is an update button, and a 'forward' image in each item of the voting list to inform that they are touchable. Second, to show the images of the institutions in which the project is developed.

- Slider. A component used to select a single value from a range of values. It is used to allow users to select the value of the vote that they wish to give to each configuration. To do this, the number of sliders implemented depends on the number of configurations in each voting. As was discussed in Section 3.1, users can define votes which have a value inside of the range [-10, 10]. In addition, a step of one is defined.

- Text. A component for displaying text. Through this component, all textual information of the application is displayed. For example, this allows the application to inform the users which configuration is the winner, which method is applied, or which preference values are defined.

- TextInput. A foundational component for inputting text into the app via a keyboard. This component is used to allow the user to introduce new values of username and password in the profile screen.

- TouchableHighlight. A wrapper for making views respond properly to touches. In addition, on press down, the opacity of the wrapped view is decreased. This component is used as a button in the application. In particular, it is used as the items in the voting list and to refresh the list of voting, to navigate to the profile screen, and to send new preferences.

- ActivityIndicator. Display a circular loading indicator. This component is useful to inform the user that the application is loading some necessary resource which requires a certain amount of time. In particular, this component is used during the requests from the application to the voting platform server that is used to obtain voter and voting information. In addition, it is also used during a load of information stored in the mobile device (private key, password and username).

- Alert. Launches an alert dialogue with a specified title and message. Using this element notifies the user that the new preferences are sent to the voting platform correctly.

All these components are implemented to develop the React application. In addition, style sheets are used, which are an abstraction similar to CSS StyleSheets. They are properties used to define the style of the components of the application. For this, attributes such as position, colour and size are defined. Figure 6.2 presents the views of the final application implemented. As can be seen, the implemented application has two main screens which allow users to use the voting platform. In addition, the application has another screen corresponding to the 'SettingsScreen' component, which enables to modify the password and user stored in the application and used in the voting platform.

The two screens shown in Figure 6.2 correspond to the 'HomeScreen' and 'VotingScreen' components. The screen on the left allows users to select a voting in which a user is registered to access it. The screen of the selected voting is then displayed, which corresponding to the screen on the right. In this way, a user can know information about a voting in which he is registered and modify his preferences for that voting.

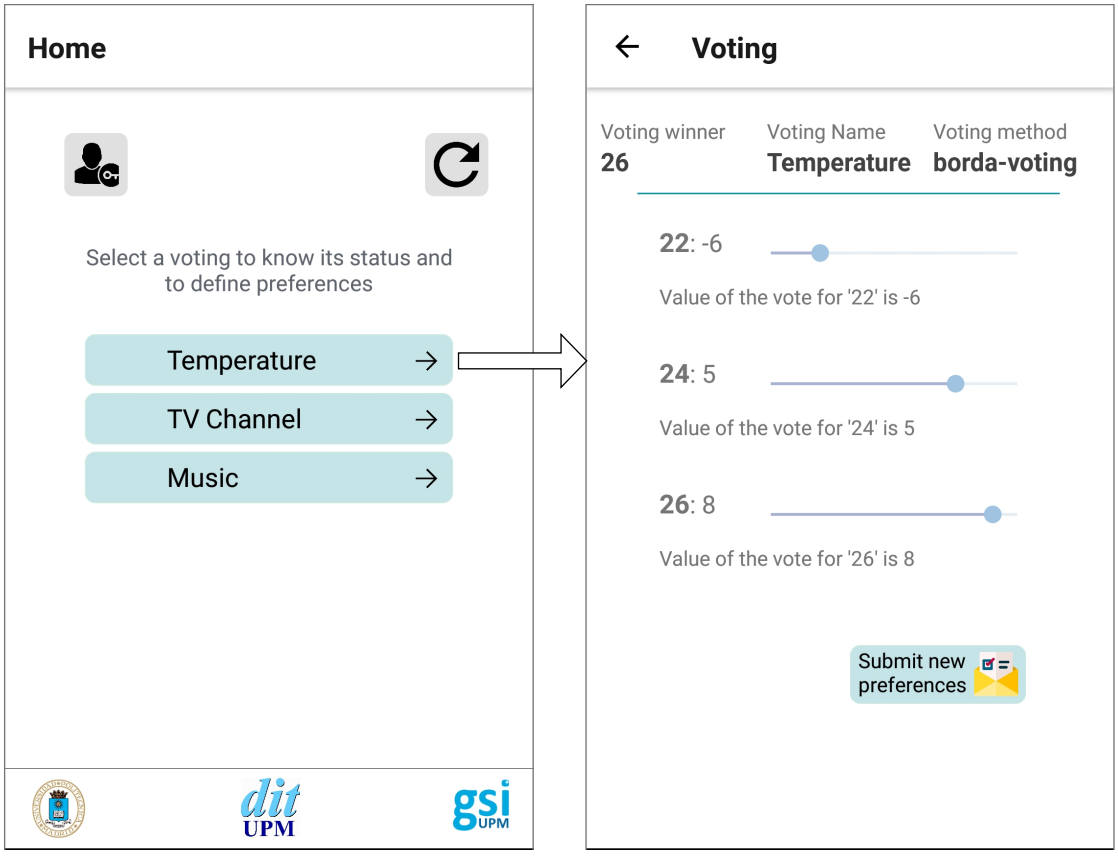

Figure 6.2: Client application of the voting platform

6.2 Methods, modules and Sawtooth SDK

The functionality of the application which satisfies the functional requirements is implemented in the components of the views. For this, three resources are used, which are: (i) methods provided by React Native, (ii) external modules or libraries, and (iii) the SDK of Sawtooth.

Two methods of React Native are mainly used. First, 'AsyncStorage', which is a simple, asynchronous, persistent, key-value storage system that is global to the app. This method can be used to store information about users inside the device, accessing it even if the device has been turned off. In particular, It is used to store the private key, password and username of the user.

The second method of React Native is 'Fetch', which enable to load resources from and send resources to remote URLs, covering networking needs. In this application, this method is used for two operations: requesting information on a voter and on votes, and updating a voter's information about their preferences. In addition to these two methods, other methods of React native used are the associated with the components life cycle, such as 'componentDidMount.

Regarding libraries, the data processing in the communication with the REST API of Sawtooth requires operations of conversion, encoding and hashing. Therefore, three libraries are used for these operations, which are 'utf8', 'js-sha512' and 'base-64'.

The Sawtooth Javascript SDK is used to establish communication with the Sawtooth API. However, a problem has been found. This SDK requires the use of 'Crypto' and 'CBOR' modules for encryption, encoding, hashing and signing. These modules produce an error in React Native. According to their official website [166], this error is due to the fact that these modules are Built-in Node modules, which do not run on devices with an environment that run JS on JavaScriptCore.

In addition, Sawtooth SDK Javascript uses the native operation 'randomBytes' for the initialisation of the private key. This operation produces an 'old browser' error in the mobile display, which does not seem to be corrected for the moment.

As a solution, there are some different no official modules which try to adapt these problematic modules for React Native. However, these modules have been tested and, generally, the response has not been positive, either for lack of support or the occurrence of new

errors. Of these no-official modules, only one could be used, 'react-native-securerandom'. This module enables to replace the 'randomBytes' operation with one that works correctly in React Native.

According to all this, a different solution is chosen. This solution is based on the use of Browserify, an open tool to bundle and adapt modules in the client using the same syntax that in Node. Therefore, any module can be used in the React Native environment as a Build-in Node module.

Browserify is used considering some web guides to implement and provide a different no-official version of the original Sawtooth SDK Javascript. This new version 'Sawtooth-SDK-React-Native' enable to create mobile applications with React Native that supports the SDK Sawtooth client.

Therefore, this new version of the SDK of Sawtooth is created considering the 'react-native-securerandom' module, the 'CBORJS' with a few modifications (adapted), and the application of Browserify tool over original Sawtooth SDK Javascript. This process is showed in Figure 6.3.

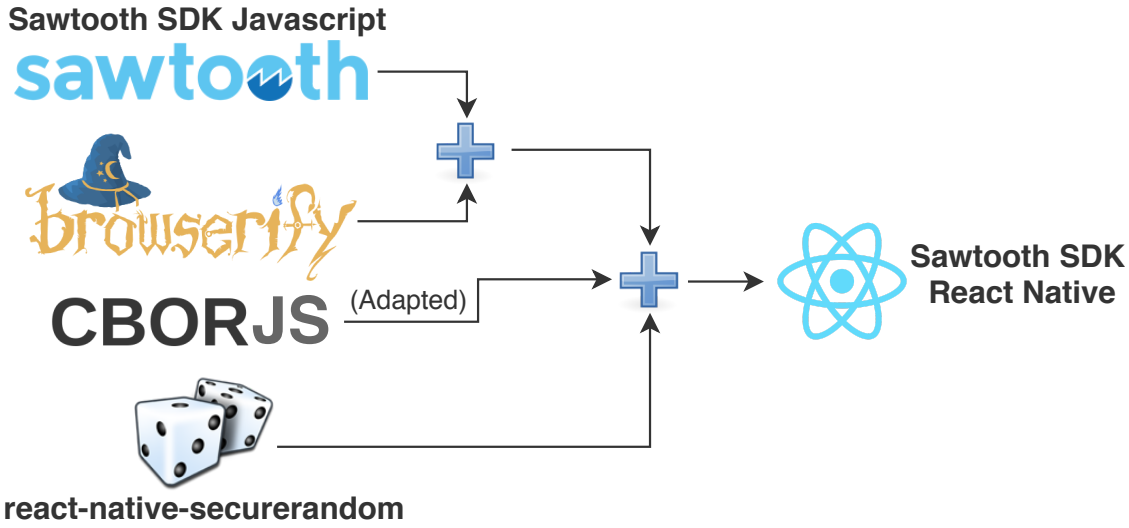

Figure 6.3: Components of Sawtooth SDK for React Native

6.3 Scenario description and client functionality implementation

The architecture of a scenario of the final system is shown in Figure 6.4. It considers the deployment of the blockchain voting platform and the smartphone application. In particular, the architecture presents four elements, which are described below.

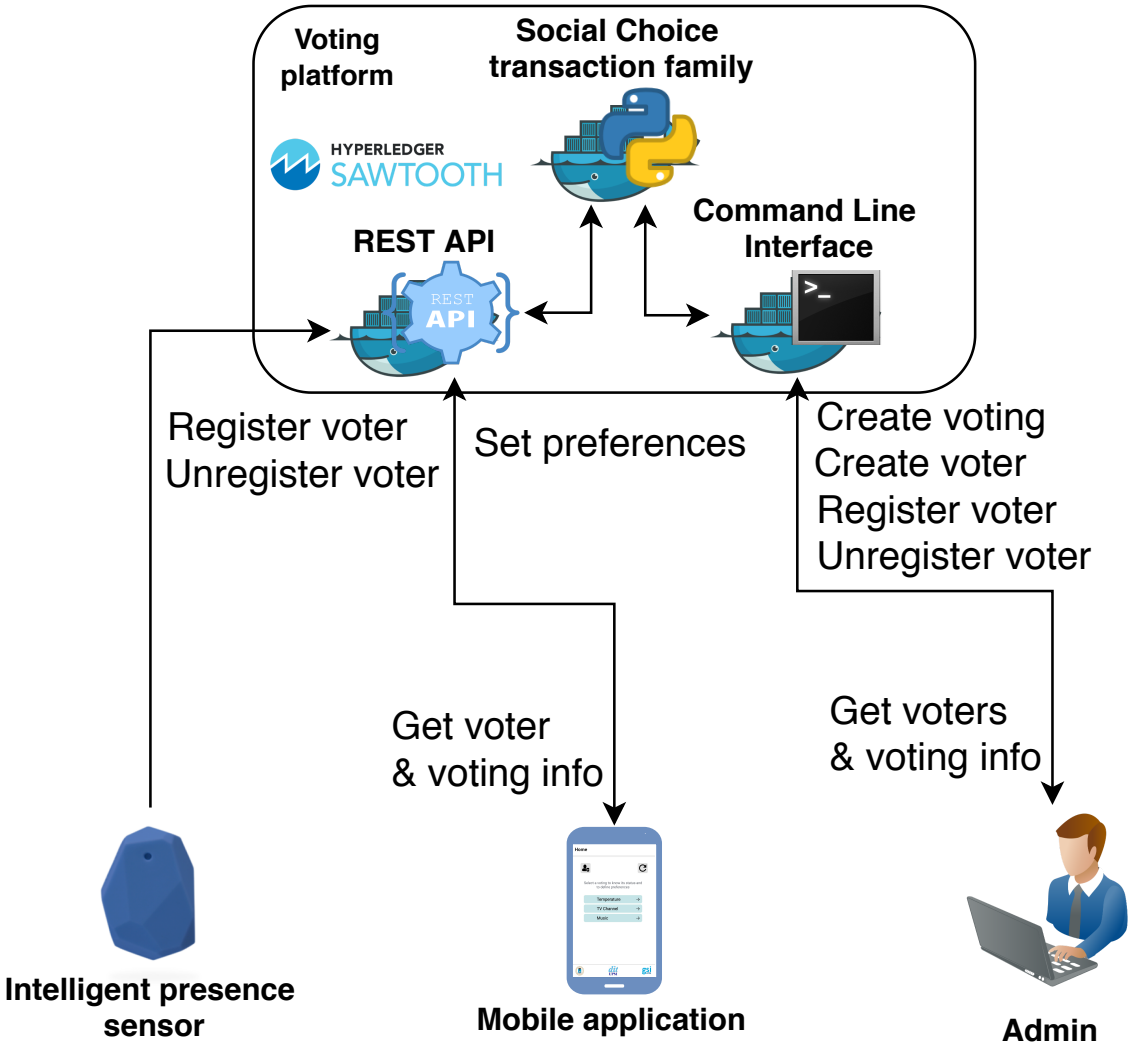

Figure 6.4: Architecture of a final scenario

The first element is the Voting platform, which is the blockchain Sawtooth system. This is deployed with the implementation of the transaction family, which applies the logic of the system: voting operations based on Social Choice methods. This element has two interfaces, which are a REST API and a Command Line Interface.

The REST API is used to communicate with the platform through JSON and GET and POST requests. The Command Line Interface enables to execute text commands. Both interfaces allow controlling the blockchain through eight possible actions. Section 5.1 presents more information on the interfaces and the transaction family.

The second element is the mobile application. This application connects to the REST interface to send two requests: get information about a voter and about a voter, and define preferences for voting.

The third element is the administrator of the system, which uses the Command Line Interface. This is the element responsible for the creation of votes and voters and for the registration and unregistration of voters from voting. Moreover, the administrator can obtain information about the voters and votes in order to manage the system.

The last element is an intelligent presence sensor. It has the ability to detect the presence of a user in a certain space, such as a room, a building or space in a city. In addition, this sensor can send REST requests to the interface of the voting platform. In this way, it can register and unregister voters from voting.

Therefore, both the administrator and the intelligent sensor can manage the registration and unregistration of voters, this depends on each particular application. For example, the administrator can be responsible for registers voter in voting which does not depend on spacial positions.

Communication between the mobile application and the REST service of the voting platform is based on three procedures. These procedures are obtaining the URLs to which to make requests, obtaining information from voters and votes, and requesting a change of preferences.

First, the URL is obtained as the union of two parts, which are the IP address and port of the REST service, the name of the transaction family. In addition, if the request is to obtain information about an instance, the address of the voter or the voting are given in the URL. Moreover, both the transaction family name and the address of the voter or voting are codified to 'UTF8' and hashed using SHA512 cryptographic function.

Second, obtaining information about voters and votes only requires applying a 'fecth' operation on that the before formed URL. If a password has been defined, it is included in the request. The answer is a JSON with a data field. This field is decoded from 'base64' and parsed to JSONs and lists.

Finally, the process of encoding and submitting information to the distributed ledger is a little more complicated. This is because a series of cryptographic safeguards are required. These are used to confirm identity and data validity. In particular, the next steps are followed.

1. Private key and signer. The first step is to create a private key if it is not created and stored on the device. For this, a secure random byte generator is used. Any set of 32 bytes can be used as the private key. It is used to confirm the identity of the sender

and sign the information that is sent to the voting platform.

This key is the only way to prove the identity of a user in the blockchain and cannot be retrieved. Therefore, it is important that its storage on the mobile device is secure. However, this aspect is considered beyond this work since efforts are focused on demonstrating the validity of the proposed solution and not on covering all aspects of a final product.

2. Payload. After creating the private key, the payload is encrypted. Transaction payloads are composed of binary-encoded data that is opaque to the blockchain. The logic for encoding and decoding depends on the transaction family definition. As a result, there are many possible formats.

   In particular, the recommendation of Sawtooth documentation is to use 'CBOR' codification. It is used to encode a payload that contains the needed information to change the voter's preferences. This data includes the name of the method for changing preferences, the name of the voter and of the voting, and a dictionary with the preferences. The format of this information should follow the format defined in the method, which is described in Section 5.1.

3. Transaction. The changes of states of the blockchain are defined in transactions. They are formed of a binary-encoded payload, which contains the data; a binary-encoded transaction header, which contains cryptographic safeguards and metadata; and a signature of that header. The cryptographic safeguards are the public keys associated with the signatures and a hash of the payload bytes.

   The metadata of the header is defined considering information for routing a transaction to the correct transaction family. This information includes input and output state addresses, references to other transactions, and the transaction family name and version.

   In this particular application, there are no dependencies with other transactions. The inputs and outputs are the state addresses that can be read and written. They are considered for reasons of efficiency and security. In this case, they are the addresses of the voter and the voting. For more information on the addresses of the state see Section 2.4.4.

   Once the transaction header is constructed, it is signed with the private key. This header signature is the ID of the transaction. Then, the header bytes, the header signature, and the payload bytes are used to form the transaction.

4. Batch. When one or more transactions are declared, they are grouped in a batch. In

this application there are no dependent transactions, so the defined batch includes only one transaction at a time. A dependent transaction is a transaction that is executed if and only if others are also executed at the same time.

However, even if a transaction is not dependent on any other, it must be submitted in a Batch. For this, a batch header is also created. This includes the public key and a list of transactions' ID, which is the signature of the transaction header. Then, the header is signed and the signature is the ID of the Batch. Finally, the Batch is formed using the header bytes, the header signature, and the list of transactions.

5. Submitting of the batches. In this application, the REST API is used to submit the batch bytes to the platform. For this, a POST request is sent to the '/batches' endpoint of the IP address and port of the REST service. This request has a body with the batch bytes and is sent using the 'fetch' method. Then, new preferences are loaded.

CHAPTER 7

Conclusions

This chapter presents three closing sections of the project. First, the objectives achieved are presented and described. Second, the most relevant conclusions about the project are presented. Finally, future work is discussed.

7.1 Achieved Goals

In this master thesis, a set of goals were established. This section presents a summary of the final outcomes in order to evaluate the fulfilment of those goals.

1. A **Social Choice model**, which enable to apply voting methods from Social Choice theory has been defined and implemented. For this, eight voting methods, an evaluation method, a historical record and some theoretical characteristics of the model has been implemented. In particular, this model has been implemented using the Python language, providing a free and open source installable software package.

2. A **simulation tool** which enables to study the case of uses related to Social Choice models in smart environments has been defined and implemented. In addition, this tool has been used to carry out a study of a proposed case of use. This case of use has been based on improving the thermal comfort of occupancy in smart buildings using voting methods. This study has provided interesting results on the usefulness of voting methods to increase the quality of life in smart environments. For the evaluation, two metrics of thermal comfort and one of satisfaction with the voting results were considered.

3. A **voting platform** based on blockchain technology has been designed and implemented. This platform enables to define voters and votes to apply methods of Social Choice theory. All information about voters, votes and results is stored in a distributed ledger. Users can use this platform through a command line interface or a REST API. This platform has been implemented using Hyperldeger Sawtooth and the deployment is based on Dockers containers.

4. An **ontology** has been defined. This ontology enables a definition of the information provided by the voting platform that is more understandable for humans and machines. Thus, it improves the use and the integration of the platform. For the definition of the ontology, OWL semantic technology has been used.

5. A **mobile application** has been designed and implemented. It allows users of an smart environment to use the voting platform. The implementation has been done using React Native. In this manner, the application is available for both iOS and Android. This application allows voters to manage their voting preferences and know information about them.

7.2 Conclusion

Smart cities can improve quality of life of the citizens satisfying their needs and demands. For this, citizens can participate in the definition of the services, which can operate considering citizens preferences and opinions. This process is the empowerment of citizens or 'democratisation'.

Smart buildings can be seen as components of smart cities. Thus, smart buildings are a interesting way to study smart city developments. Both usually considers the Internet of Things and Ambient Intelligence technologies to provide a smart environment, which can be aware of the preferences and actions of the people. It enable to improve the well-being of the people.

Usually, the smart environments provide shared services. In this situation, they should consider preferences from multiple users and use this information to maximise their welfare. A solution is the consideration of Social Choice theory, which defines voting methods to obtain a overall social preferences from combination of individuals preferences.

Therefore, a Social Choice model is defined and implemented in this work. This model is developed considering different features and components. The features consider aspects of Social Choice theory such as the modelling of votes or the application of randomisation techniques.

The components enable to apply operations required to transform preferences in a ranking of winners. This is, they are used to apply voting methods. For example, filtering, fitting or calculation operations are considered. In addition, other components enable to calculate evaluation metrics of voting methods and a record of historical results. In particular, eight voting methods have been implemented.

Approaching an Internet of Things and Ambient Intelligence deployment is quite attractive in order to implements and experiment with different service models. However, this involve considerable economic and time costs and has risks related to the validity and usefulness of a considered service model. Because of this, considering a simulation model is a good alternative, which enables to study applications in smart environments without these drawbacks.

Thus, a model of simulation is design and implemented with the goal of studying a case of use based on applying voting methods to improve the thermal comfort of occupants in

a smart building. In particular, a Multi-Agent System is implemented, which is a suitable combination of social science and computing. It is constituted by occupation and HVAC systems agents. The behaviour of occupants agents is modelled considering non-homogeneous Markov chains and schedules. In addition, each occupant has a preference of temperature.

Moreover, a physical thermal model is implemented in the simulation model, which enables to model the temperature. For this, thermal zones and the heat balance method are considered. Comfort evaluation methods are used to know the occupant comfort with the temperature in a room. For this, two methods are considered, which are the Fanger's method and a method proposed based on thermal perception. In addition, the satisfaction of the occupants with the voting method is also considered.

An experiment is carried out simulating a real scenario to study the eigth proposed voting methods. This has obtained four main interesting results, which are the following. First, considering methods such as voting methods which enable to adapt services to the preferences of the people increases their comfort and satisfaction. Second, the definition of evaluation methods of comfort which considers individual preferences are useful to evaluate services.

Third, allowing voters to grade all the voting candidates produces more representative results of the collective. Fourth, the best voting methods are those which consider more information about the preferences of voters. In addition, the modelling of votes must be done according to the voting method chosen, avoiding to produce noise or to waste information.

A solution which enables to democratises services in smart environments is the deployment of platforms. They allow citizens to participate in voting or debating processes to reflect their opinion and preferences. Thus, platforms can consider the preferences of the citizens to apply methods which enable increase their quality of life.

In this work, a voting platform is implemented. For this, blockchain technology is used. Blockchain has been proposed to voting related systems such as E-voting since it provides secure communication platforms. In particular, the voting platform is implemented using Hyperledger Sawtooth.

The platform is implemented considering multiple identical nodes forming a network. Each node has six main components and a copy of the blockchain. Each of these components is defined accordingly to the voting application. They have specific function, such as work as clients of the application, manage the settings and apply consensus algorithms.

In particular, there is a main component which provides the application functionality. This component defines the data model of the application, applies the operations and manages the business logic. Therefore, it integrates the Social Choice model.

In this manner, a platform which allows users to apply voting methods on the configuration of services and store the information securely in a blockchain is provided. For its use, six different actions have been implemented, which enable to apply operations such as defining a voter, configuring preferences and applying voting methods.

In order to move the information provided by the voting platform from data to semantic concepts an ontology is defined. It increases the precision and efficiency in the use of information, promoting greater consistency and understanding of the meaning of information to both humans and computing systems.

The proposed ontology considers seven classes, twelve relations and five data properties to define concepts used by the voting platform. Thus, this ontology considers aspects concerning to the Social Choice model. For this, no ontology already defined has been found. In particular, the proposed ontology is used to define a knowledge exchange system based on a REST API.

Smartphones are a important tool of smart environments, which enable that services can be autonomously adjusted in real time. They can interact with the smart environments using an IP connection over a data-link service. Therefore, in addition to the platform, a mobile application which allows users to use the voting platform is implemented in this work.

This application is implemented using React Native framework, which enables to deploy a application in both iOS and Android operative systems. The development of the application is based on the use of a set of UI components and the implementation of client functionality.

For this, one of the requirements is to use a Sawtooth module. This led to compatibility problems, which are solved using additional tools. In this way, a client which allows users to reflect their preferences to the voting platform and know information about votes is provided. In addition, a possible final deployment scenario is described.

7.3 Future work

This project has completed tasks related to several technological lines. Each one of these lines can be continued with future work. Some aspects of this future work have already been named throughout this document as work beyond this project. Others are related to new paths different from the ones considered in this project. Some of the tasks which can be completed as future work are presented below. This has the goal of motivating growth of the development and research work carried out in this project. In order to better representation and description of this future work, the task is presented are ordered items. In particular, ten tasks have been proposed, which are the following.

1. Implementation and deployment of a proxy component for the voting platform. This task is based on the implementation and integration of a proxy in the voting platform. For this, the Sawtooth SDK could be used. This proxy enables to manage the authentication levels of the platform.

2. Use of the ontology proposed in this work in the REST service of the voting platform. The future task to integrate the proposed ontology in the proxy or REST service of the platform. This enables to provide various advantages, which has been commented in this work, such as improved understanding and integration.

3. Deployment of a version of the platform with multiple validating nodes. The future task to replicate the deployment made in this work in multiple nodes. The competition of this task is required for deployments in real environments.

4. Definition of a private key generation method for the voting platform. This task considers the definition and implementation of a mechanism for the definition of a private user key according to an identification method. This method can be controlled by the user, such as an electronic ID, or a mechanism managed by an administrator, such as previous physical registrations. This task enables to provide more security to the platform and facilitate the identification of participants, which is really interesting in a voting system.

5. Extension of the Social Choice model. This future work considers the extension of the voting model through the consideration of new voting methods and new evaluation measures. In addition to this, it is proposed as an interesting future work the consideration of 'unfair' participants, who apply voting strategies that involve defining false preferences. For this, it is proposed as an option the use of a voting record, which enable to evaluate when users modify their preferences suspiciously.

6. Definition of new case studies. In this project, several case studies related to smart cities and smart buildings have been discussed. In particular, an application related to the HVAC system in an intelligent building has been considered. Therefore, this future task aims to propose new use cases, which can be studied in the simulation tool.

7. Physical deployment of the solution (complete or partial). This future task enables to test the proposed solution in a real scenario to assess it and generate new future tasks. As a possible initial scenario, the deployment is proposed in a set of rooms. Later, it can be extended to a building and finally to a city.

8. Testing of the mobile application with real users. This task enables to obtain feedback from real users to improve the functional and non-functional requirements of the mobile application and voting system.

9. Use of other technologies of smart environments. There are situations in which voters do not directly provide their preferences, either due to incapacity or idleness. In this situation, a future task is to use monitoring technologies, such as sentiments detection using facial recognition. It enables to know the level of satisfaction of all users, even when they do not participate in the voting. Other technologies such as machine learning algorithms also can be used.

10. Application of the solution in a political E-Voting application. As a future task, it is proposed to study the use of the voting platform in an electronic voting application for political elections. This future task is considered really interesting because of its social value. In fact, some countries such as Estonia have already considered these systems. Moreover, these systems enable to expand the concept of democracy towards a more participatory society.

APPENDIX A

Ethical, economic, social and environmental aspects

This appendix presents the possible ethical, economic, social and environmental impacts associated with this work. Therefore, this section approach aspect related to the responsibility of practical application of engineering, such as social and environmental impact, commitment to professional ethics, responsibility, legality and standards of the practical application of engineering.

A.1 Introduction

This paper has proposed a solution to implement and deploy a voting platform. This platform allows users in an intelligent space to vote, transmitting their preferences on the possible configurations of services and systems. In particular, an intelligent building has been considered as the case of use. However, the potential use of these voting systems in other uses, such as smart cities and e-voting systems, has been also discussed in this work.

In addition to the voting platform, a mobile application has been designed and implemented. It can be used as a client of the voting platform, allowing users to vote. In this way, the users' smartphone can communicate with sensors deployed in the intelligent environment, which know if a user is in a location. Thus, the sensors can register them in a vote, allowing them to vote.

Before the implementation of the voting platform and the mobile application, a study of voting methods have been carried out using social simulation techniques. All of these aspects have a set of ethical, social, economic and environmental implications, which are discussed below.

A.2 Ethical and professional implications

The voting platform implemented in this project stores user data. Although these data are not necessarily identifying since they do not consider real names but nicknames or usernames, the fact that in the voting platform users reflect a preference must consider. This preference can be, according to its purpose, sensitive information. Therefore, the processing of these data must provide guarantees to users, relating to the use of data and privacy. In fact, security aspects are essential in voting systems, and this consideration has determined which technology has been chosen to implement the solution.

Moreover, the voting process involves a temporary monitoring mechanism. This information, about when someone did something must also be protected. This is related to another aspect to consider, which is that the system of registering voters in a vote can be done by sensors. These sensors can know information about the actions of the users in intelligent space. The storage of this information and the delimitation of its use is also an aspect that must be considered.

Finally, the application of a study using social simulation has required knowledge about

the activity of people in a building. This data has been obtained anonymously and informing the participants about the purpose of the survey. All before named aspects related to data must comply with the 'European General Data Protection Regulation 2016/679'.

Professional aspects related to intellectual property rights do not apply since all the software used is open source and free license.

A.3 Social impact

The aim of this work is to achieve a strong social impact. Although effort has been dedicated to performing a study based on simulations, the ultimate goal of this work is the deployment of a system usable by people. Moreover, this system is designed, implemented and deployed to improve the quality of life of people. In particular, it aims to improve the lives of all citizens residing in cities with a smart environment.

The improvement of citizens' lives is achieved through empowerment. This is because citizens are provided with a way to express their opinion and increase their participation in public and also private systems and services. This can enhance the quality of the systems and services provided at the same time as institutions and organisations obtain greater knowledge about their users.

With a view to the future, the system proposed in this paper can favour a democratisation of society. It can support democratic systems and institutions, favouring greater participation, a service of greater security and integrity and a means of increasing the number of decisions in which citizens can participate.

There are problems related to e-accessibility. They can affect elderly people and those who by personal choice or incapacity have difficulties to use smartphones and mobile applications. Although the usability of the system has been taken into account, it requires a minimum of familiarisation with the technology and the availability of a smartphone. This problem can generate social discrimination and social isolation.

Although a functional solution has been proposed, this work is focused on future generations. Thus, it is considered that most of the problems will be mitigated. However, alternative solutions for specific situations, such as disabilities, must be considered. In this case, the best solution is to work with public institutions to provide adequate alternatives.

Considering other social aspects, the proposed system provides mechanisms of non-

discrimination and non-judgement, where users are considered with full equality.

A.4 Economic impact

This project proposes an implementation of a system that allows people to express their opinion on systems and services. This can not only enhance the quality of the systems and services provided but also provide valuable information to institutions and organizations. This information can be transformed into costs reduction and improved integration and social acceptance of innovation.

In addition, knowing people's preferences also enables the development of new business models. Moreover, the improvement of people's quality of life is related to an improvement in the performance of tasks and labour productivity. Furthermore, if the system is used for an e-voting solution, it can reduce the costs required for voting.

On the other hand, it is necessary to consider the costs associated with the deployment and maintenance of the system. These costs will be higher when the solution is bigger. However, technologies such as cloud computing can be used to improve performance and reduce costs.

A.5 Environmental impact

This work has three main environmental impacts. The first one is related to the batteries of the smartphones since they are used by the users of the system. The second corresponds to the energy consumption of the system deployment, which can be lower if solutions such as cloud computing are considered.

The third and most important one is related to the sensors of the intelligent environment. These sensors must be, when possible, connected to the power line. However, it is not always possible and batteries will be required. In addition, these sensors will eventually be replaced.

Therefore, it is necessary to consider solutions related to batteries, such as making them as durable as possible and with a lower contamination impact. In addition, it is necessary to use sensors of minimum consumption. Moreover, It is interesting to use sensors manufactured with recyclable materials. In this project, a blockchain solution which reduces energy consumption has been considered.

APPENDIX B

Economic budget

This appendix presents the economic budget required to carry out this project. In particular, costs related to material, licences and human resources are described.

B.1 Material resources and licences

In order to carry out this project, the following material resources have been used. First, a personal computer which has been used for all tasks, both for the design and implementation of the mobile application and voting platform and for the implementation and use of the simulation tool. Therefore, it has also been used for the study of voting models. Second, a smartphone with Android operating system that has been used to test the mobile application. In particular, the cost of a personal computer was approximately 500 euros. The cost of the smartphone was 200 euros.

The voting service could be deployed in a cloud service to test the deployment in a real-world scenario. In this case, it is necessary to consider the costs of a Dockers container deployment service, such as Amazon Web Services. Prices vary depending on the provider and the extension of the service. A cost of 40 euros per month is considered as a minimum requirement for testing the deployed service.

Moreover, the economic cost could consider other aspects of the deployment such as sensors. However, this aspect is considered beyond the scope of this project.

The need to acquire licenses is not considered since all the software and tools used are free and open-source.

B.2 Human resources

Considering the work of one person, approximate time of seven months working part-time is required to carry out this project. This consideration includes the time employed in the study of the state of the art and tools, the designing, developing and testing of the systems, and the obtaining and analysis of simulation results.

According to this, the salary of an engineering graduate part-time working is considered. In particular, a salary of 600 euros is considered. Therefore, the cost of human resources is of approximately 600x7 euros per month, which is 4.200 euros in total. As can be seen, this cost does not include the post-project maintenance and operation. Moreover, if a deployment is made, the costs of installation personnel should be considered, which beyond the scope of this project.

APPENDIX C

Installation, deployment and configuration of the systems

This appendix presents considerations related to the systems developed in this project. In this way, a description of aspects related to installation, use and additional documentation is presented. In particular, this appendix consists of four sections, which present the Social Choice model software package, the simulation software tool, the voting blockchain platform and the mobile application.

The belief that the free and open-source software is a fundamental complement to research works in computer sciences is considered in this work. This is because it enables to verify results and to perform new research. In addition, providing the software enables new lines of work to extend, reuse and improve it. Therefore, all software implemented in this project is provided as free and open source software.

C.1 Social Choice Model

This software enables to use of a Social Choice model. In particular, it provides voting methods, mechanisms for filtering and fitting of votes, and evaluation methods based on satisfaction with the winners.

This software has been called SocePy (**So**cial Choi**ce** in **Py**thon). This is implemented in Python and is provided as an installable Pip package. This way, the package can be installed using the next command: 'pip install socepy'. A Python 3 version or higher is required for installation and use.

The source code, a test file of the methods and additional information is provided in the Github repository[1] of the package.

C.2 Modelling and Simulation tool

This tool enables to simulate models oriented to people in spaces. In particular, the tool has been implemented to model occupants in a smart building. In addition, it models electrical systems, such as HVAC systems or lighting. This tool has been implemented using Python and packages such as Mesa and Transaction. A Python 3 version or higher is required for installation and use.

The source code and additional information about the software are provided in the Github repository[2] of the project. Moreover, detailed documentation is provided in Read the Docs[3].

[1] https://github.com/gsi-upm/SocePy
[2] https://github.com/gsi-upm/soba
[3] https://soba.readthedocs.io/en/latest/

C.3 Voting platform

The voting platform enables to apply Social Choice theory methods remotely and record the information in a blockchain. Sawtooth SDK Python has been used for the implementation. More about this SDK is described in the official Sawtooth documentation[4].

The implemented voting platform is provided in a github repository[5]. The Docker tool and Docker-compose are required for its use. Information about its installation can be found on the official websites[6] [7].

The deployment of the platform is based on the source code provided in the repository and some configuration and deployment files. The main configuration file is used by the Docker tool to launch the platform. In particular, the configuration file is a Docker-compose.yaml, which is presented below.

```
version: '3.6'

services:

  settings-tp:
    image: hyperledger/sawtooth-settings-tp:nightly
    container_name: sawtooth-settings-tp
    depends_on:
      - validator
    command: |
      bash -c "settings-tp -vv -C tcp://validator:4004"
    stop_signal: SIGKILL

  soce-tp-python:
    build:
      context: .
      dockerfile: examples/soce_python/Dockerfile
      args:
        - http_proxy
        - https_proxy
        - no_proxy
    image: soce-tp-python-local:${ISOLATION_ID}
    volumes:
```

[4]https://sawtooth.hyperledger.org/docs/core/releases/1.1.5/
[5]https://github.com/gsi-upm/sawtooth-soce
[6]https://docs.docker.com/v17.09/engine/installation/
[7]https://docs.docker.com/compose/install/

```
    - ./:/project/sawtooth-sdk-python
  container_name: soce-tp-python-local
  depends_on:
    - validator
  command: |
    bash -c "
      bin/protogen
      cd examples/soce_python
      python3 setup.py clean --all
      python3 setup.py build
      soce-tp-python -vv -C tcp://validator:4004
    "
  stop_signal: SIGKILL

client:
  image: hyperledger/sawtooth-shell:nightly
  container_name: sawtooth-shell
  depends_on:
    - validator
  command: |
    bash -c "sawtooth keygen && tail -f /dev/null"
  volumes:
    - ./bin:/home/sawtooth/bin
    - ./examples/soce_python/sawtooth_soce:/home/sawtooth/sawtooth_soce
    - ./examples/soce_python/sawtooth_soce:/usr/lib/python3/dist-packages
  /sawtooth_soce
    - ./bin/soce-tp-python:/usr/bin/soce-tp-python
    - ./bin/soce:/usr/bin/soce
  stop_signal: SIGKILL

validator:
  image: hyperledger/sawtooth-validator:nightly
  container_name: sawtooth-validator
  expose:
    - 4004
    - 8800
    - 5050
  ports:
    - "4004:4004"
  command: |
    bash -c "
      sawadm keygen
      sawset genesis \
        -k /etc/sawtooth/keys/validator.priv \
        -o config-genesis.batch && \
```

```
      sawset proposal create \
        -k /etc/sawtooth/keys/validator.priv \
        sawtooth.consensus.algorithm.name=Devmode \
        sawtooth.consensus.algorithm.version=0.1 \
        -o config.batch && \
      sawadm genesis config-genesis.batch config.batch && \
      sawtooth-validator -vv \
          --endpoint tcp://validator:8800 \
          --bind component:tcp://eth0:4004 \
          --bind network:tcp://eth0:8800 \
          --bind consensus:tcp://eth0:5050 \
    "
  stop_signal: SIGKILL

rest-api:
  image: hyperledger/sawtooth-rest-api:nightly
  container_name: sawtooth-rest-api
  ports:
    - "8008:8008"
  depends_on:
    - validator
  command: |
    bash -c "sawtooth-rest-api -v --connect tcp://validator:4004 --bind
  rest-api:8008"
  stop_signal: SIGKILL

devmode-rust:
  image: hyperledger/sawtooth-devmode-engine-rust:nightly
  container_name: sawtooth-devmode-engine-rust
  depends_on:
    - validator
  command: |
    bash -c "devmode-engine-rust -v --connect tcp://validator:5050"
  stop_signal: SIGKILL
```

Accordingly, two steps are required for the deployment of the voting platform, which are as follows. First, download source code from the Github repository: 'git clone https://github.com/gsi-upm/sawtooth-soce'. Second, deployment of the platform using the configuration file: 'docker-compose -f sawtooth-soce/docker-compose.yaml up'.

C.4 Mobile application

This application is provided to be used as a client of the voting platform. For its implementation, React Native has been used. Therefore, it is required for use. The official website[8] describes the steps for its installation.

The execution of the application can be done using a browser-based client (called Expo) or with Android studio. Both options are explained in React Native website.

The source code of the application is provided in a Github repository[9]. In this repository is also the Sawtooth module for React Native implemented in this project.

[8] https://github.com/gsi-upm/react-native-soce
[9] https://facebook.github.io/react-native/

APPENDIX D

Ontology proposed

In this appendix, the ontology proposed in this work is presented. In addition, this appendix presents an example of use of this ontology.

D.1 Definition of the ontology proposed

Section 5.3 presents an ontology, which has been proposed to define the information provided by the REST API of the voting platform. The complete definition of this ontology is presented below in Turtle format.

```
@prefix : <http://www.semanticweb.org/merinom/ontologies/2019/4/untitled-
    ontology-30#> .
@prefix dbp: <http://dbpedia.org/page/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix soce: <http://www.gsi.dit.upm.es/ontologies/soce#> .
@base <http://www.semanticweb.org/merinom/ontologies/2019/4/untitled-
    ontology-30> .

#################################################################
#    Classes
#################################################################

###  http://dbpedia.org/page/Social_Choice
dbp:Social_Choice rdf:type owl:Class .

###  http://dbpedia.org/page/Voter
dbp:Voter rdf:type owl:Class .

###  http://dbpedia.org/page/Voting
dbp:Voting rdf:type owl:Class .

###  http://dbpedia.org/page/Voting_method
dbp:Voting_method rdf:type owl:Class .

###  http://www.gsi.dit.upm.es/ontologies/soce#Configuration
soce:Configuration rdf:type owl:Class .

###  http://www.gsi.dit.upm.es/ontologies/soce#Preference
soce:Preference rdf:type owl:Class .

###  http://www.gsi.dit.upm.es/ontologies/soce#Voting_result
soce:Voting_result rdf:type owl:Class .
```

```
#################################################################
#    Object Properties
#################################################################

###  http://www.gsi.dit.upm.es/ontologies/soce#appliesVotingMethod
soce:appliesVotingMethod rdf:type owl:ObjectProperty ;
                         rdfs:domain dbp:Voting ;
                         rdfs:range dbp:Voting_method .

###  http://www.gsi.dit.upm.es/ontologies/soce#considersVoter
soce:considersVoter rdf:type owl:ObjectProperty ;
                    rdfs:domain dbp:Social_Choice ;
                    rdfs:range dbp:Voter .

###  http://www.gsi.dit.upm.es/ontologies/soce#definesPreference
soce:definesPreference rdf:type owl:ObjectProperty ;
                       rdfs:domain dbp:Voter ;
                       rdfs:range soce:Preference .


###  http://www.gsi.dit.upm.es/ontologies/soce#hasConfiguration
soce:hasConfiguration rdf:type owl:ObjectProperty ;
                      rdfs:domain dbp:Voting ;
                      rdfs:range soce:Configuration .

###  http://www.gsi.dit.upm.es/ontologies/soce#hasFirstPlaceWinner
soce:hasFirstPlaceWinner rdf:type owl:ObjectProperty ;
                         rdfs:domain soce:Voting_result ;
                         rdfs:range soce:Configuration .


###  http://www.gsi.dit.upm.es/ontologies/soce#hasSecondPlaceWinner
soce:hasSecondPlaceWinner rdf:type owl:ObjectProperty ;
                          rdfs:domain soce:Voting_result ;
                          rdfs:range soce:Configuration .

###  http://www.gsi.dit.upm.es/ontologies/soce#hasThirdPlaceWinner
soce:hasThirdPlaceWinner rdf:type owl:ObjectProperty ;
                         rdfs:domain soce:Voting_result ;
                         rdfs:range soce:Configuration .


###  http://www.gsi.dit.upm.es/ontologies/soce#isRegisteredInVoting
soce:isRegisteredInVoting rdf:type owl:ObjectProperty ;
```

```
                          rdfs:domain dbp:Voter ;
                          rdfs:range dbp:Voting .

###  http://www.gsi.dit.upm.es/ontologies/soce#obtainsVotingResult
soce:obtainsVotingResult rdf:type owl:ObjectProperty ;
                         rdfs:domain dbp:Voting ;
                         rdfs:range soce:Voting_result .

###  http://www.gsi.dit.upm.es/ontologies/soce#performsVoting
soce:performsVoting rdf:type owl:ObjectProperty ;
                    rdfs:domain dbp:Social_Choice ;
                    rdfs:range dbp:Voting .

###  http://www.gsi.dit.upm.es/ontologies/soce#referencesVoting
soce:referencesVoting rdf:type owl:ObjectProperty ;
                      rdfs:domain soce:Preference ;
                      rdfs:range dbp:Voting .

###  http://www.gsi.dit.upm.es/ontologies/soce#valuesConfiguration
soce:valuesConfiguration rdf:type owl:ObjectProperty ;
                         rdfs:domain soce:Preference ;
                         rdfs:range soce:Configuration .

#################################################################
#    Data properties
#################################################################

###  http://www.gsi.dit.upm.es/ontologies/soce#configurationValue
soce:configurationValue rdf:type owl:DatatypeProperty ;
                        rdfs:domain soce:Configuration ;
                        rdfs:range xsd:string .

###  http://www.gsi.dit.upm.es/ontologies/soce#methodName
soce:methodName rdf:type owl:DatatypeProperty ;
                rdfs:domain dbp:Voting_method ;
                rdfs:range xsd:string .

###  http://www.gsi.dit.upm.es/ontologies/soce#preferenceValue
soce:preferenceValue rdf:type owl:DatatypeProperty ;
                     rdfs:domain soce:Preference ;
                     rdfs:range xsd:integer .

###  http://www.gsi.dit.upm.es/ontologies/soce#voterName
soce:voterName rdf:type owl:DatatypeProperty ;
               rdfs:domain dbp:Voter ;
```

```
                rdfs:range xsd:string .

###  http://www.gsi.dit.upm.es/ontologies/soce#votingName
soce:votingName rdf:type owl:DatatypeProperty ;
                rdfs:domain dbp:Voting ;
                rdfs:range xsd:string .
```

D.2 Example using the ontology

An example using the proposed ontology is presented below. In particular, this example considers a situation in which a voter participates in two votes: a selection of music type and temperature. Each voting has three possible configurations. The voter determines the value of a vote for each configuration, which is the preference. In this way, each voting gets a result with the ranking of the winners. This example is presented in JSON-LD format.

```
{
  "@context": {
    "owl": "http://www.w3.org/2002/07/owl#",
    "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
    "rdfs": "http://www.w3.org/2000/01/rdf-schema#",
    "soce": "http://www.gsi.dit.upm.es/ontologies/soce#",
    "xsd": "http://www.w3.org/2001/XMLSchema#"
  },
  "@graph": [
    {
      "@id": "soce:Voting_Music",
      "@type": [
        "http://dbpedia.org/page/Voting",
        "owl:NamedIndividual"
      ],
      "soce:appliesVotingMethod": {
        "@id": "soce:Voting_Method_Borda"
      },
      "soce:hasConfiguration": [
        {
          "@id": "soce:Configuration_Music_2"
        },
        {
          "@id": "soce:Configuration_Music_3"
```

```
        },
        {
          "@id": "soce:Configuration_Music_1"
        }
      ],
      "soce:obtainsVotingResult": {
        "@id": "soce:Voting_Result_1"
      },
      "soce:votingName": "Music"
    },
    {
      "@id": "soce:Voter_1",
      "@type": [
        "owl:NamedIndividual",
        "http://dbpedia.org/page/Voter"
      ],
      "soce:definesPreference": [
        {
          "@id": "soce:Preference_Music_3"
        },
        {
          "@id": "soce:Preference_Music_2"
        },
        {
          "@id": "soce:Preference_Music_1"
        },
        {
          "@id": "soce:Preference_Temperature_1"
        },
        {
          "@id": "soce:Preference_Temperature_3"
        },
        {
          "@id": "soce:Preference_Temperature_2"
        }
      ],
      "soce:isRegisteredInVoting": [
        {
          "@id": "soce:Voting_Temperature"
        },
        {
          "@id": "soce:Voting_Music"
        }
      ],
      "soce:voterName": "VoterUsername"
```

```
  },
  {
    "@id": "soce:Social_Choice_Building",
    "@type": [
      "owl:NamedIndividual",
      "http://dbpedia.org/page/Social_Choice"
    ],
    "soce:considersVoter": {
      "@id": "soce:Voter_1"
    },
    "soce:performsVoting": [
      {
        "@id": "soce:Voting_Temperature"
      },
      {
        "@id": "soce:Voting_Music"
      }
    ]
  },
  {
    "@id": "soce:Preference_Music_1",
    "@type": [
      "owl:NamedIndividual",
      "soce:Preference"
    ],
    "soce:preferenceValue": 0,
    "soce:referencesVoting": {
      "@id": "soce:Voting_Music"
    },
    "soce:valuesConfiguration": {
      "@id": "soce:Configuration_Music_1"
    }
  },
  {
    "@id": "soce:Preference_Music_3",
    "@type": [
      "owl:NamedIndividual",
      "soce:Preference"
    ],
    "soce:preferenceValue": 8,
    "soce:referencesVoting": {
      "@id": "soce:Voting_Music"
    },
    "soce:valuesConfiguration": {
      "@id": "soce:Configuration_Music_3"
```

```
    }
  },
  {
    "@id": "soce:Preference_Music_2",
    "@type": [
      "owl:NamedIndividual",
      "soce:Preference"
    ],
    "soce:preferenceValue": 4,
    "soce:referencesVoting": {
      "@id": "soce:Voting_Music"
    },
    "soce:valuesConfiguration": {
      "@id": "soce:Configuration_Music_2"
    }
  },
  {
    "@id": "soce:Preference_Temperature_1",
    "@type": [
      "owl:NamedIndividual",
      "soce:Preference"
    ],
    "soce:preferenceValue": -6,
    "soce:referencesVoting": {
      "@id": "soce:Voting_Temperature"
    },
    "soce:valuesConfiguration": {
      "@id": "soce:Configuration_Temperature_1"
    }
  },
  {
    "@id": "soce:Preference_Temperature_2",
    "@type": [
      "soce:Preference",
      "owl:NamedIndividual"
    ],
    "soce:preferenceValue": 5,
    "soce:referencesVoting": {
      "@id": "soce:Voting_Temperature"
    },
    "soce:valuesConfiguration": {
      "@id": "soce:Configuration_Temperature_2"
    }
  },
  {
```

```
    "@id": "http://www.semanticweb.org/merinom/ontologies/2019/4/untitled
  -ontology-30",
    "@type": "owl:Ontology"
  },
  {
    "@id": "soce:Preference_Temperature_3",
    "@type": [
      "owl:NamedIndividual",
      "soce:Preference"
    ],
    "soce:preferenceValue": 8,
    "soce:referencesVoting": {
      "@id": "soce:Voting_Temperature"
    },
    "soce:valuesConfiguration": {
      "@id": "soce:Configuration_Temperature_3"
    }
  },
  {
    "@id": "soce:Voting_Result_2",
    "@type": [
      "soce:Voting_result",
      "owl:NamedIndividual"
    ],
    "soce:hasFirstPlaceWinner": {
      "@id": "soce:Configuration_Temperature_3"
    },
    "soce:hasSecondPlaceWinner": {
      "@id": "soce:Configuration_Temperature_2"
    },
    "soce:hasThirdPlaceWinner": {
      "@id": "soce:Configuration_Temperature_1"
    }
  },
  {
    "@id": "soce:Configuration_Temperature_2",
    "@type": [
      "soce:Configuration",
      "owl:NamedIndividual"
    ],
    "soce:configurationValue": 24
  },
  {
    "@id": "soce:Voting_Result_1",
    "@type": [
```

```
        "soce:Voting_result",
        "owl:NamedIndividual"
      ],
      "soce:hasFirstPlaceWinner": {
        "@id": "soce:Configuration_Music_3"
      },
      "soce:hasSecondPlaceWinner": {
        "@id": "soce:Configuration_Music_2"
      },
      "soce:hasThirdPlaceWinner": {
        "@id": "soce:Configuration_Music_1"
      }
    },
    {
      "@id": "soce:Voting_Temperature",
      "@type": [
        "owl:NamedIndividual",
        "http://dbpedia.org/page/Voting"
      ],
      "soce:appliesVotingMethod": {
        "@id": "soce:Voting_Method_Borda"
      },
      "soce:hasConfiguration": [
        {
          "@id": "soce:Configuration_Temperature_1"
        },
        {
          "@id": "soce:Configuration_Temperature_3"
        },
        {
          "@id": "soce:Configuration_Temperature_2"
        }
      ],
      "soce:obtainsVotingResult": {
        "@id": "soce:Voting_Result_2"
      },
      "soce:votingName": "Temperature"
    },
    {
      "@id": "soce:Voting_Method_Borda",
      "@type": [
        "owl:NamedIndividual",
        "http://dbpedia.org/page/Voting_method"
      ],
      "soce:methodName": "borda-voting"
```

```
    },
    {
      "@id": "soce:Configuration_Music_2",
      "@type": [
        "owl:NamedIndividual",
        "soce:Configuration"
      ],
      "soce:configurationValue": "Pop"
    },
    {
      "@id": "soce:Configuration_Music_1",
      "@type": [
        "soce:Configuration",
        "owl:NamedIndividual"
      ],
      "soce:configurationValue": "Rock"
    },
    {
      "@id": "soce:Configuration_Temperature_1",
      "@type": [
        "soce:Configuration",
        "owl:NamedIndividual"
      ],
      "soce:configurationValue": 22
    },
    {
      "@id": "soce:Configuration_Music_3",
      "@type": [
        "soce:Configuration",
        "owl:NamedIndividual"
      ],
      "soce:configurationValue": "ChillOut"
    },
    {
      "@id": "soce:Configuration_Temperature_3",
      "@type": [
        "owl:NamedIndividual",
        "soce:Configuration"
      ],
      "soce:configurationValue": 26
    }
  ]
}
```

Bibliography

[1] Anthony Simonofski, Estefanía Serral Asensio, Johannes De Smedt, and Monique Snoeck. Citizen participation in smart cities: Evaluation framework proposal. In *2017 IEEE 19th conference on business informatics (CBI)*, volume 1, pages 227–236. IEEE, 2017.

[2] Saraju P Mohanty, Uma Choppali, and Elias Kougianos. Everything you wanted to know about smart cities: The internet of things is the backbone. *IEEE Consumer Electronics Magazine*, 5(3):60–70, 2016.

[3] Robert G Hollands. Will the real smart city please stand up? intelligent, progressive or entrepreneurial? *City*, 12(3):303–320, 2008.

[4] Daniel Belanche, Luis V Casaló, and Carlos Orús. City attachment and use of urban services: Benefits for smart cities. *Cities*, 50:75–81, 2016.

[5] Lasse Steenbock Vestergaard, João Fernandes, and Mirko Alexander Presser. Towards smart city democracy. *Geoforum Perspektiv*, 14(25), 2015.

[6] Karim Benouaret, Raman Valliyur-Ramalingam, and François Charoy. Crowdsc: Building smart cities with large-scale citizen participation. *IEEE Internet Computing*, 17(6):57–63, 2013.

[7] Jalpa Shah and Biswajit Mishra. Iot enabled environmental monitoring system for smart cities. In *2016 International Conference on Internet of Things and Applications (IOTA)*, pages 383–388. IEEE, 2016.

[8] Andrea Zanella, Nicola Bui, Angelo Castellani, Lorenzo Vangelista, and Michele Zorzi. Internet of things for smart cities. *IEEE Internet of Things journal*, 1(1):22–32, 2014.

[9] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future generation computer systems*, 29(7):1645–1660, 2013.

[10] Carlos Ramos, Juan Carlos Augusto, and Daniel Shapiro. Ambient intelligence—the next step for artificial intelligence. *IEEE Intelligent Systems*, 23(2):15–18, 2008.

[11] Michael O'grady and Gregory O'hare. How smart is your city? *Science*, 335(6076):1581–1582, 2012.

[12] Diane J Cook, Juan C Augusto, and Vikramaditya R Jakkula. Ambient intelligence: Technologies, applications, and opportunities. *Pervasive and Mobile Computing*, 5(4):277–298, 2009.

[13] Judith Masthoff, Wamberto W Vasconcelos, Chris Aitken, and FS Correa da Silva. Agent-based group modelling for ambient intelligence. In *AISB symposium on affective smart environments, Newcastle, UK*, 2007.

[14] Emile Aarts and Reiner Wichert. Ambient intelligence. In *Technology guide*, pages 244–249. Springer, 2009.

[15] Emilio Serrano, Pablo Moncada, Mercedes Garijo, and Carlos A Iglesias. Evaluating social choice techniques into intelligent environments by agent based social simulation. *Information Sciences*, 286:102–124, 2014.

[16] Tobias Nipkow. Social choice theory in hol. *Journal of Automated Reasoning*, 43(3):289–304, 2009.

[17] Piotr Krzysztof Skowron and Edith Elkind. Social choice under metric preferences: scoring rules and stv. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[18] Michał Pawlak, Jakub Guziur, and Aneta Poniszewska-Marańda. Voting process with blockchain technology: auditable blockchain voting system. In *International Conference on Intelligent Networking and Collaborative Systems*, pages 233–244. Springer, 2018.

[19] Simona Ibba, Andrea Pinna, Matteo Seu, and Filippo Eros Pani. Citysense: blockchain-oriented smart cities. In *Proceedings of the XP2017 Scientific Workshops*, page 12. ACM, 2017.

[20] Ibrar Yaqoob, Ibrahim Abaker Targio Hashem, Yasir Mehmood, Abdullah Gani, Salimah Mokhtar, and Sghaier Guizani. Enabling communication technologies for smart cities. *IEEE Communications Magazine*, 55(1):112–120, 2017.

[21] Saeed Karshenas and Mehrdad Niknam. Ontology-based building information modeling. In *Computing in Civil Engineering (2013)*, pages 476–483. 2013.

[22] Lu Tan, Mingyuan Hu, and Hui Lin. Agent-based simulation of building evacuation: Combining human behavior with predictable spatial accessibility in a fire emergency. *Information Sciences*, 295:53–66, 2015.

[23] Xiang Li, Gang Liu, Anhong Ling, Jian Zhan, Ning An, Lian Li, and Yongzhong Sha. Building a practical ontology for emergency response systems. In *Computer Science and Software Engineering, 2008 International Conference on*, volume 4, pages 222–225. IEEE, 2008.

[24] Alberto Vanolo. Is there anybody out there? the place and role of citizens in tomorrow's smart cities. *Futures*, 82:26–36, 2016.

[25] Zhu Wang, Lingfeng Wang, Anastasios I Dounis, and Rui Yang. Integration of plug-in hybrid electric vehicles into energy and comfort management for smart building. *Energy and Buildings*, 47:260–266, 2012.

[26] Madhavi Indraganti and Kavita Daryani Rao. Effect of age, gender, economic group and tenure on thermal comfort: a field study in residential buildings in hot and dry climate with seasonal variations. *Energy and buildings*, 42(3):273–281, 2010.

[27] Giancarlo Fortino, Wilma Russo, and Claudio Savaglio. Agent-oriented modeling and simulation of iot networks. In *2016 federated conference on computer science and information systems (FedCSIS)*, pages 1449–1452. IEEE, 2016.

[28] Eric Bonabeau. Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences*, 99(suppl 3):7280–7287, 2002.

[29] Vito Fragnelli, Giovanni Monella, and Guido Ortona. A simulative approach for evaluating electoral systems. *Homo Oeconomicus*, 22(4):525–549, 2005.

[30] Maxime Héroux-Legault. Using electoral simulations to study voting behavior. 2019.

[31] Jianjun Cheng, Xinhong Yin, Qi Li, Haijuan Yang, Longjie Li, Mingwei Leng, and Xiaoyun Chen. Voting simulation based agglomerative hierarchical method for network community detection. *Scientific reports*, 8(1):8064, 2018.

[32] Kaja J Fietkiewicz, Agnes Mainka, and Wolfgang G Stock. egovernment in cities of the knowledge society. an empirical investigation of smart cities' governmental websites. *Government Information Quarterly*, 34(1):75–83, 2017.

[33] Angelo Cenedese, Andrea Zanella, Lorenzo Vangelista, and Michele Zorzi. Padova smart city: An urban internet of things experimentation. In *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, pages 1–6. IEEE, 2014.

[34] Giuseppe Piro, Ilaria Cianci, Luigi Alfredo Grieco, Gennaro Boggia, and Pietro Camarda. Information centric services in smart cities. *Journal of Systems and Software*, 88:169–188, 2014.

[35] Engel J Knibbe. Building management system, October 15 1996. US Patent 5,565,855.

[36] Pervez Hameed Shaikh, Nursyarizal Bin Mohd Nor, Perumal Nallagownden, Irraivan Elamvazuthi, and Taib Ibrahim. A review on optimized control systems for building energy and comfort management of smart sustainable buildings. *Renewable and Sustainable Energy Reviews*, 34:409–429, 2014.

[37] Laura Klein, Jun-young Kwak, Geoffrey Kavulya, Farrokh Jazizadeh, Burcin Becerik-Gerber, Pradeep Varakantham, and Milind Tambe. Coordinating occupant behavior for building energy and comfort management using multi-agent systems. *Automation in construction*, 22:525–536, 2012.

[38] Sami Karjalainen. Gender differences in thermal comfort and use of thermostats in everyday thermal environments. *Building and environment*, 42(4):1594–1603, 2007.

[39] Pervez Hameed Shaikh, Nursyarizal Bin Mohd Nor, Perumal Nallagownden, Irraivan Elamvazuthi, and Taib Ibrahim. A review on optimized control systems for building energy and comfort management of smart sustainable buildings. *Renewable and Sustainable Energy Reviews*, 34:409–429, 2014.

[40] Sarah Mennicken, Jo Vermeulen, and Elaine M Huang. From today's augmented houses to tomorrow's smart homes: new directions for home automation research. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 105–115. ACM, 2014.

[41] Jorg P Muller and Jörg P Müller. *The design of intelligent agents: a layered approach*, volume 1177. Springer Science & Business Media, 1996.

[42] Sascha Ossowski. *Agreement technologies*, volume 8. Springer Science & Business Media, 2012.

[43] Haradhan Mohajan. Social welfare and social choice in different individuals' preferences. 2011.

[44] Kenneth J Arrow. *Social choice and individual values*, volume 12. Yale university press, 2012.

[45] Kenneth J Arrow. A difficulty in the concept of social welfare. *Journal of political economy*, 58(4):328–346, 1950.

[46] Alan D Taylor. *Social choice and the mathematics of manipulation*. Cambridge University Press, 2005.

[47] Bojan Srdjevic. Linking analytic hierarchy process and social choice methods to support group decision-making in water management. *Decision Support Systems*, 42(4):2261–2273, 2007.

[48] Andre d'Angelo, Abdollah Eskandari, and Ferenc Szidarovszky. Social choice procedures in water-resource management. *Journal of Environmental Management*, 52(3):203–210, 1998.

[49] David Klahr. A computer simulation of the paradox of voting. *American Political Science Review*, 60(2):384–390, 1966.

[50] Hannu Nurmi. An assessment of voting system simulations. *Public Choice*, 73(4):459–487, 1992.

[51] Gordon Tullock and Colin D Campbell. Computer simulation of a small voting system. *The Economic Journal*, 80(317):97–104, 1970.

[52] Paul E Johnson. Simulation modeling in political science. *American Behavioral Scientist*, 42(10):1509–1530, 1999.

[53] Dennis C Mueller, Geoffrey C Philpotts, and Jaroslav Vanek. The social gains from exchanging votes: A simulation approach. *Public Choice*, 13(1):55–79, 1972.

[54] Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D Procaccia. *Handbook of computational social choice*. Cambridge University Press, 2016.

[55] Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D Procaccia. Introduction to computational social choice, 2016.

[56] Craig Boutilier, Britta Dorn, Nicolas Maudet, Vincent Merlin, et al. Computational social choice: Theory and applications. Technical report, 2015.

[57] Markus Brill, Rupert Freeman, and Vincent Conitzer. Computing the optimal game. 2015.

[58] Sanna Laukkanen, Annika Kangas, and Jyrki Kangas. Applying voting theory in natural resource management: a case of multiple-criteria group decision support. *Journal of Environmental Management*, 64(2):127–137, 2002.

[59] Haris Aziz, Felix Brandt, Edith Elkind, and Piotr Skowron. Computational social choice: The first ten years and beyond. *Computer science today*, 10000, 2017.

[60] Ali M Aseere, Enrico H Gerding, and David E Millard. A voting-based agent system for course selection in e-learning. In *2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, volume 2, pages 303–310. IEEE, 2010.

[61] LA Hurtado, PH Nguyen, WL Kling, and W Zeiler. Building energy management systems—optimization of comfort and energy use. In *2013 48th International Universities' Power Engineering Conference (UPEC)*, pages 1–6. IEEE, 2013.

[62] JT Oden, T Belytschko, TJR Hughes, C Johnson, D Keyes, A Laub, L Petzold, D Srolovitz, and S Yip. Revolutionizing engineering science through simulation: A report of the national science foundation blue ribbon panel on simulation-based engineering science. *Arlington, VA: National Science Foundation*, 2006.

[63] Xiaohang Feng, Da Yan, and Tianzhen Hong. Simulation of occupancy in buildings. *Energy and Buildings*, 87:348–359, 2015.

[64] Mengda Jia, Ravi S Srinivasan, and Adeeba A Raheem. From occupancy to occupant behavior: An analytical survey of data acquisition technologies, modeling methodologies and simulation coupling mechanisms for building energy efficiency. *Renewable and Sustainable Energy Reviews*, 68:525–540, 2017.

[65] Rafael H Bordini, Lars Braubach, Mehdi Dastani, A El F Seghrouchni, Jorge J Gomez-Sanz, Joao Leite, Gregory O'Hare, Alexander Pokahr, and Alessandro Ricci. A survey of programming languages and platforms for multi-agent systems. *Informatica*, 30(1), 2006.

[66] Xuan Luo, Khee Poh Lam, Yixing Chen, and Tianzhen Hong. Performance evaluation of an agent-based occupancy simulation model. *Building and Environment*, 115:42–53, 2017.

[67] Jie Zhao, Bertrand Lasternas, Khee Poh Lam, Ray Yun, and Vivian Loftness. Occupant behavior and schedule modeling for building energy simulation through office appliance power consumption data mining. *Energy and Buildings*, 82:341–355, 2014.

[68] Da Yan, William O'Brien, Tianzhen Hong, Xiaohang Feng, H Burak Gunay, Farhang Tahmasebi, and Ardeshir Mahdavi. Occupant behavior modeling for building performance simulation: Current state and future challenges. *Energy and Buildings*, 107:264–278, 2015.

[69] Rui Yang and Lingfeng Wang. Development of multi-agent system for building energy and comfort management based on occupant behaviors. *Energy and Buildings*, 56:1–7, 2013.

[70] W Kreuter and H Hofmann. Electrolysis: the important energy transformer in a world of sustainable energy. *International Journal of Hydrogen Energy*, 23(8):661–666, 1998.

[71] Hai-xiang Zhao and Frédéric Magoulès. A review on the prediction of building energy consumption. *Renewable and Sustainable Energy Reviews*, 16(6):3586–3592, 2012.

[72] Daniel Coakley, Paul Raftery, and Marcus Keane. A review of methods to match building energy simulation models to measured data. *Renewable and sustainable energy reviews*, 37:123–141, 2014.

[73] Abdul Afram and Farrokh Janabi-Sharifi. Review of modeling methods for hvac systems. *Applied Thermal Engineering*, 67(1):507–519, 2014.

[74] Richard Strand, Fred Winkelmann Fred Buhl, Joe Huang, and Russell Taylor. Enhancing and extending the capabilities of the building heat balance simulation technique for use in energyplus. In *in Proceedings of Building Simulation'99, Volume II*, pages 653–660. IBPSA, International Building Performance Simulation Association, 1999.

[75] Paul Davidsson and Magnus Boman. Distributed monitoring and control of office buildings by embedded agents. *Information Sciences*, 171(4):293–307, 2005.

[76] Justin R Dobbs and Brandon M Hencey. Model predictive hvac control with online occupancy model. *Energy and Buildings*, 82:675–684, 2014.

[77] R Chargui and H Sammouda. Modeling of a residential house coupled with a dual source heat pump using trnsys software. *Energy Conversion and Management*, 81:384–399, 2014.

[78] Drury B Crawley, Linda K Lawrie, Frederick C Winkelmann, Walter F Buhl, Y Joe Huang, Curtis O Pedersen, Richard K Strand, Richard J Liesen, Daniel E Fisher, Michael J Witte, et al. Energyplus: creating a new-generation building energy simulation program. *Energy and buildings*, 33(4):319–331, 2001.

[79] Steve Sorrell et al. The rebound effect: an assessment of the evidence for economy-wide energy savings from improved energy efficiency, 2007.

[80] Fabrizio Ascione, Nicola Bianco, Claudio De Stasio, Gerardo Maria Mauro, and Giuseppe Peter Vanoli. Simulation-based model predictive control by the multi-objective optimization of building energy performance and thermal comfort. *Energy and Buildings*, 111:131–144, 2016.

[81] Varick L Erickson and Alberto E Cerpa. Occupancy based demand response hvac control strategy. In *Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building*, pages 7–12. ACM, 2010.

[82] Satoshi Nakamoto et al. Bitcoin: A peer-to-peer electronic cash system. 2008.

[83] J Leon Zhao, Shaokun Fan, and Jiaqi Yan. Overview of business innovations and research opportunities in blockchain and introduction to the special issue, 2016.

[84] Marten Risius and Kai Spohrer. A blockchain research framework. *Business & Information Systems Engineering*, 59(6):385–409, 2017.

[85] Emre Yavuz, Ali Kaan Koç, Umut Can Çabuk, and Gökhan Dalkılıç. Towards secure e-voting using ethereum blockchain. In *2018 6th International Symposium on Digital Forensic and Security (ISDFS)*, pages 1–7. IEEE, 2018.

[86] Fran Casino, Thomas K Dasaklis, and Constantinos Patsakis. A systematic literature review of blockchain-based applications: current status, classification and open issues. *Telematics and Informatics*, 2018.

[87] D Drescher. Blockchain basics: A non-technical introduction in 25 steps, 1st edn. apress, frankfurt am main (2017).

[88] G Karame and E Audroulaki. Bitcoin and blockchain security. artech house inc., norwood (2016).

[89] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151:1–32, 2014.

[90] Vikram Dhillon, David Metcalf, and Max Hooper. The hyperledger project. In *Blockchain enabled applications*, pages 139–149. Springer, 2017.

[91] Hl_whitepaper_introductiontohyperledger.pdf. `https://www.hyperledger.org/wp-content/uploads/2018/07/HL_Whitepaper_IntroductiontoHyperledger.pdf`. (Accessed on 05/14/2019).

[92] Hyperledger_arch_wg_paper_2_smartcontracts.pdf. `https://www.hyperledger.org/wp-content/uploads/2018/04/Hyperledger_Arch_WG_Paper_2_SmartContracts.pdf`. (Accessed on 05/14/2019).

[93] behlendorf_hyperledgerslides. `http://www.redwoodmednet.org/projects/events/20160718/docs/rwmn_20160718_behlendorf.pdf`. (Accessed on 05/14/2019).

[94] Kelly Olson, Mic Bowman, James Mitchell, Shawn Amundson, Dan Middleton, and Cian Montgomery. Sawtooth: An introduction. *The Linux Foundation, Jan*, 2018.

[95] Architecture guide — sawtooth v1.1.5 documentation. `https://sawtooth.hyperledger.org/docs/core/releases/latest/architecture.html`. (Accessed on 05/31/2019).

[96] Cristiano De Faveri, Ana Moreira, João Araújo, and Vasco Amaral. Towards security modeling of e-voting systems. In *2016 IEEE 24th International Requirements Engineering Conference Workshops (REW)*, pages 145–154. IEEE, 2016.

[97] Renming Qi, Chen Feng, Zheng Liu, and Nezih Mrad. Blockchain-powered internet of things, e-governance and e-democracy. In *E-Democracy for Smart Cities*, pages 509–520. Springer, 2017.

[98] Feasibility study on internet voting for the central electoral commission of the republic of moldova: Report and preliminary roadmap. Technical report, United Nations Development Programm, 2016.

[99] Areejessa/hyperledger-composer-voting-app. `https://github.com/AreejEssa/Hyperledger-Composer-Voting-App`. (Accessed on 05/30/2019).

[100] html5-ninja/hyperledger-vote-app: Blockchain digital vote web app built with hyperledger composer. https://github.com/html5-ninja/hyperledger-vote-app. (Accessed on 05/30/2019).

[101] Bitagora: A decentralized voting platform built on hyperledger sawtooth – hyperledger. https://www.hyperledger.org/blog/2018/09/20/bitagora-a-decentralized-voting-platform-built-on-hyperledger-sawtooth. (Accessed on 05/30/2019).

[102] Natalya Fridman Noy and Carole D Hafner. The state of the art in ontology design: A survey and comparative review. *AI magazine*, 18(3):53, 1997.

[103] Tom R Gruber. Knowledge acquisition. *A translation approach to portable ontologies*, 5(2):199–220, 1993.

[104] Franz Baader, Diego Calvanese, Deborah McGuinness, Peter Patel-Schneider, and Daniele Nardi. *The description logic handbook: Theory, implementation and applications*. Cambridge university press, 2003.

[105] Yasir Javed, Tony Norris, and David Johnston. Ontology-based inference to enhance team situation awareness in emergency management. In *Proceedings of the 8th International ISCRAM Conference*, pages 1–9, 2011.

[106] Thomas R Gruber. Toward principles for the design of ontologies used for knowledge sharing? *International journal of human-computer studies*, 43(5-6):907–928, 1995.

[107] Shang-Hsien Hsieh, Hsien-Tang Lin, Nai-Wen Chi, Kuang-Wu Chou, and Ken-Yu Lin. Enabling the development of base domain ontology through extraction of knowledge from engineering domain handbooks. *Advanced Engineering Informatics*, 25(2):288–296, 2011.

[108] Margrethe Kobes, Ira Helsloot, Bauke De Vries, and Jos G Post. Building safety and human behaviour in fire: A literature review. *Fire Safety Journal*, 45(1):1–11, 2010.

[109] Le Zhang and Raja RA Issa. Ontology-based partial building information model extraction. *Journal of Computing in Civil Engineering*, 27(6):576–584, 2012.

[110] Janez Brank, Marko Grobelnik, and Dunja Mladenić. A survey of ontology evaluation techniques. 2005.

[111] Robert Arp, Barry Smith, and Andrew D Spear. *Building ontologies with basic formal ontology*. Mit Press, 2015.

[112] Mariano Rodrıguez-Muro, Josef Hardi, and Diego Calvanese. Quest: efficient sparql-to-sql for rdf and owl. In *11th International Semantic Web Conference ISWC*, page 53. Citeseer, 2012.

[113] Rizwan Iqbal, Masrah Azrifah Azmi Murad, Aida Mustapha, and Nurfadhlina Mohd Sharef. An analysis of ontology engineering methodologies: A literature review. *Research journal of applied sciences, engineering and technology*, 6(16):2993–3000, 2013.

[114] Mariano Fernández-López and Asunción Gómez-Pérez. Overview and analysis of methodologies for building ontologies. *The Knowledge Engineering Review*, 17(2):129–156, 2002.

[115] Oscar Corcho, Mariano Fernández-López, and Asunción Gómez-Pérez. Methodologies, tools and languages for building ontologies. where is their meeting point? *Data & knowledge engineering*, 46(1):41–64, 2003.

[116] Tom Gruber. What is an ontology. *WWW Site http://www-ksl. stanford. edu/kst/whatis-an-ontology. html (accessed on 07-09-2004)*, 1993.

[117] Mike Uschold. Building ontologies: Towards a unified methodology. In *Proceedings of 16th Annual Conference of the British Computer Society Specialists Group on Expert Systems*. Citeseer, 1996.

[118] Julita Bermejo. A simplified guide to create an ontology. *Madrid University*, pages 1–12, 2007.

[119] Natalya F Noy, Deborah L McGuinness, et al. Ontology development 101: A guide to creating your first ontology, 2001.

[120] Mark Chignell, James Cordy, Joanna Ng, and Yelena Yesha. *The smart internet: current research and future applications*, volume 6400. Springer, 2010.

[121] Morton E Winston, Roger Chaffin, and Douglas Herrmann. A taxonomy of part-whole relations. *Cognitive science*, 11(4):417–444, 1987.

[122] Rebekka Volk, Julian Stengel, and Frank Schultmann. Building information modeling (bim) for existing buildings—literature review and future needs. *Automation in construction*, 38:109–127, 2014.

[123] Michael Uschold and Martin King. Towards a methodology for building ontologies. 1995.

[124] Wei Wang, Suparna De, Ralf Toenjes, Eike Reetz, and Klaus Moessner. A comprehensive ontology for knowledge representation in the internet of things. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on*, pages 1793–1798. IEEE, 2012.

[125] Xiao Hang Wang, D Qing Zhang, Tao Gu, and Hung Keng Pung. Ontology based context modeling and reasoning using owl. In *Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on*, pages 18–22. Ieee, 2004.

[126] Xin Liu, Zhongfu Li, and Shaohua Jiang. Ontology-based representation and reasoning in building construction cost estimation in china. *Future Internet*, 8(3):39, 2016.

[127] Ian Horrocks, Peter F Patel-Schneider, and Frank Van Harmelen. From shiq and rdf to owl: The making of a web ontology language. *Web semantics: science, services and agents on the World Wide Web*, 1(1):7–26, 2003.

[128] Jeff Heflin et al. Web ontology language (owl) use cases and requirements. *W3C working draft*, 31, 2003.

[129] Jeremy J Carroll and Graham Klyne. Resource description framework ({RDF}): Concepts and abstract syntax. 2004.

[130] Aditya Kalyanpur, Bijan Parsia, Evren Sirin, Bernardo Cuenca Grau, and James Hendler. Swoop: A web ontology editing browser. *Web Semantics: Science, Services and Agents on the World Wide Web*, 4(2):144–153, 2006.

[131] Teresa Onorati, Alessio Malizia, Paloma Díaz, and Ignacio Aedo. Modeling an ontology on accessible evacuation routes for emergencies. *Expert Systems with Applications*, 41(16):7124–7134, 2014.

[132] Zhijun Zhang and John A Miller. Ontology query languages for the semantic web. *A Performance Evaluation.//Department of Computer Science, University of Georgia, USA*, 2005.

[133] Eric Prud, Andy Seaborne, et al. Sparql query language for rdf. 2006.

[134] William Danielsson. React native application development: A comparison between native android and react native, 2016.

[135] Andre Charland and Brian Leroux. Mobile application development: web vs. native. *Communications of the ACM*, 54(5):49–53, 2011.

[136] Adam Boduch. *React and React Native*. Packt Publishing Ltd, 2017.

[137] Matias Martinez and Sylvain Lecomte. Towards the quality improvement of cross-platform mobile applications. In *2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*, pages 184–188. IEEE, 2017.

[138] Bonnie Eisenman. *Learning react native: building native mobile apps with javascript.* " O'Reilly Media, Inc.", 2015.

[139] Steven J. Brams and Richard F. Potthoff. The paradox of grading systems. *Public Choice*, 165(3/4):193–210, 2015.

[140] Edith Elkind and Martin Lackner. Structure in dichotomous preferences. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[141] José Carlos R Alcantud and Annick Laruelle. Dis&approval voting: a characterization. *Social Choice and Welfare*, 43(1):1–10, 2014.

[142] Freerk Auke Lootsma and H Schuijt. The multiplicative ahp, smart and electre in a common context. *Journal of Multi-Criteria Decision Analysis*, 6(4):185–196, 1997.

[143] Alan D Taylor and Allison M Pacelli. *Mathematics and politics: strategy, voting, power, and proof*. Springer Science & Business Media, 2008.

[144] Jerry S Kelly. *Social choice theory: An introduction*. Springer Science & Business Media, 2013.

[145] Piotr Faliszewski, Piotr Skowron, Arkadii Slinko, and Nimrod Talmon. Multiwinner voting: A new challenge for social choice theory. *Trends in computational social choice*, 74, 2017.

[146] Ariel D Procaccia and Jeffrey S Rosenschein. The distortion of cardinal preferences in voting. In *International Workshop on Cooperative Information Agents*, pages 317–331. Springer, 2006.

[147] Chris Wells, Justin Reedy, John Gastil, and Carolyn Lee. Information distortion and voting choices: The origins and effects of factual beliefs in initiative elections. *Political Psychology*, 30(6):953–969, 2009.

[148] Elliot Anshelevich, Onkar Bhardwaj, Edith Elkind, John Postl, and Piotr Skowron. Approximating optimal social choice under metric preferences. *Artificial Intelligence*, 264:27–51, 2018.

[149] Christian Klamler. The copeland rule and condorcet's principle. *Economic Theory*, 25(3):745–749, 2005.

[150] Mark Allen Satterthwaite. Strategy-proofness and arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *Journal of economic theory*, 10(2):187–217, 1975.

[151] Allan Gibbard et al. Manipulation of voting schemes: a general result. *Econometrica*, 41(4):587–601, 1973.

[152] John Nash. Non-cooperative games. *Annals of mathematics*, pages 286–295, 1951.

[153] William V Gehrlein. Condorcet's paradox. *Theory and Decision*, 15(2):161–197, 1983.

[154] Marquis de Condorcet. Essay on the application of analysis to the probability of majority decisions. *Paris: Imprimerie Royale*, 1785.

[155] James M Buchanan and Gordon Tullock. The calculus of consent (vol. 3). *Ann Arbor: University of Michigan Press*, pages 1–270, 1962.

[156] Refrigerating American Society of Heating and Air-Conditioning Engineers. *2001 ASHRAE Handbook: Fundamentals*. Ashrae Handbook Fundamentals Systems-International Metric System. ASHRAE, 2001.

[157] A. Bhatia. A guide to heating & cooling load estimation. Technical report, PDH Center, 2012.

[158] Ph.D. Kyoo Dong Song. Are1024 – building equipment and system design. Technical report, School of Architecture and Architectural Engineering, Hanyang University, ERICA Campus, 2005.

[159] Povl Ole Fanger. *Thermal comfort analysis and applications in environment engineering*. McGraw Hill New York, 1972.

[160] Poul O Fanger et al. Thermal comfort. analysis and applications in environmental engineering. *Thermal comfort. Analysis and applications in environmental engineering.*, 1970.

[161] ISO 7730. Ergonomics of the thermal environment. Technical report, ISO/TC 159/SC 5 Ergonomics of the physical environment, 2005.

[162] A Pharo Gagge, JAJ Stolwijk, and JD Hardy. Comfort and thermal sensations and associated physiological responses at various ambient temperatures. *Environmental research*, 1(1):1–20, 1967.

[163] J Fergus Nicol and Michael A Humphreys. Adaptive thermal comfort and sustainable thermal standards for buildings. *Energy and buildings*, 34(6):563–572, 2002.

[164] David Masad and Jacqueline Kazil. Mesa: An agent-based modeling framework. In *14th Python in Science Conference*, pages 53–60, 2015.

[165] gsi-upm/soba: Simulation occupancy based on agents. `https://github.com/gsi-upm/soba`. (Accessed on 06/14/2019).

[166] Javascript environment · react native. `https://facebook.github.io/react-native/docs/javascript-environment.html#javascript-runtime`. (Accessed on 06/14/2019).

[167] Jan Willemson. Bits or paper: Which should get to carry your vote? *Journal of information security and applications*, 38:124–131, 2018.

[168] John Conti, Paul Holtberg, Jim Diefenderfer, Angelina LaRose, James T Turnure, and Lynn Westfall. International energy outlook 2016 with projections to 2040. Technical report, USDOE Energy Information Administration (EIA), Washington, DC (United States). Office of Energy Analysis, 2016.

[169] Henrik Tommerup and Svend Svendsen. Energy savings in danish residential building stock. *Energy and buildings*, 38(6):618–626, 2006.

[170] Hashem Akbari, S Konopacki, and Melvin Pomerantz. Cooling energy savings potential of reflective roofs for residential and commercial buildings in the united states. *Energy*, 24(5):391–407, 1999.

[171] Nisha Patet Sensharma, James E Woods, and Anna K Goodwin. Relationships between the indoor environment and productivity: a literature review. *Ashrae Transactions*, 104:686, 1998.

[172] Andrea Saltelli, Karen Chan, E Marian Scott, et al. *Sensitivity analysis*, volume 1. Wiley New York, 2000.

[173] Jon Herman and Will Usher. Salib: an open-source python library for sensitivity analysis. *The Journal of Open Source Software*, 2(9), 2017.

[174] Max D Morris. Factorial sampling plans for preliminary computational experiments. *Technometrics*, 33(2):161–174, 1991.

[175] Zheng Yang and Burcin Becerik-Gerber. How does building occupancy influence energy efficiency of hvac systems? *Energy Procedia*, 88:775–780, 2016.

[176] Baizhan Li and Runming Yao. Urbanisation and its impact on building energy consumption and efficiency in china. *Renewable Energy*, 34(9):1994–1998, 2009.

[177] Eduardo Merino. soba/soba at master · gsi-upm/soba. `https://github.com/gsi-upm/soba/tree/master/soba`.

[178] Xiaoshan Pan, Charles S Han, Ken Dauber, and Kincho H Law. A multi-agent based framework for the simulation of human and social behaviors during emergency evacuations. *Ai & Society*, 22(2):113–132, 2007.

[179] Xiaomu Luo, Tong Liu, Baihua Shen, Liwen Gao, Xiaoyan Luo, et al. Human indoor localization based on ceiling mounted pir sensor nodes. In *Consumer Communications & Networking Conference (CCNC), 2016 13th IEEE Annual*, pages 868–874. IEEE, 2016.

[180] Fawwaz T Ulaby, Richard K Moore, and Adrian K Fung. *Microwave Remote Sensing, Active and Passive, Volume I, Microwave Remote Sensing Fundamentals and Radiometry*. Reading, MA: Addison-Wesley, 1986.

[181] Theis Heidmann Pedersen, Kasper Ubbe Nielsen, and Steffen Petersen. Method for room occupancy detection based on trajectory of indoor climate sensor data. *Building and Environment*, 115:147–156, 2017.

[182] Bing Dong, Burton Andrews, Khee Poh Lam, Michael Höynck, Rui Zhang, Yun-Shang Chiou, and Diego Benitez. An information technology enabled sustainability test-bed (itest) for occupancy detection through an environmental sensing network. *Energy and Buildings*, 42(7):1038–1046, 2010.

[183] Nagender Kumar Suryadevara, Subhas Chandra Mukhopadhyay, Sean Dieter Tebje Kelly, and Satinder Pal Singh Gill. Wsn-based smart sensors and actuator for power management in intelligent buildings. *IEEE/ASME Transactions On Mechatronics*, 20(2):564–571, 2015.

[184] Antonio Guerrieri, Giancarlo Fortino, Antonio Ruzzelli, and Gregory MP O'Hare. A wsn-based building management framework to support energy-saving applications in buildings. In *Advancements in Distributed Computing and Internet Technologies: Trends and Issues*, pages 258–273. IGI Global, 2012.

[185] Deborah Snoonian. Smart buildings. *IEEE spectrum*, 40(8):18–23, 2003.

[186] A Ioannou and LCM Itard. Energy performance and comfort in residential buildings: Sensitivity for building parameters and occupancy. *Energy and Buildings*, 92:216–233, 2015.

[187] Astrid Roetzel, Aris Tsangrassoulis, and Udo Dietrich. Impact of building design and occupancy on office comfort and energy performance in different climates. *Building and environment*, 71:165–175, 2014.

[188] Turismo y Comercio Ministerio de Industria. Energy in spain (in spanish). Technical report, Ministerio de Industria, Turismo y Comercio, 2006.

[189] Turismo y Comercio Ministerio de Industria. *La Energía en España*. División de información, documentación y publicaciones, 2006.

[190] Ernst Worrell, Lenny Bernstein, Joyashree Roy, Lynn Price, and Jochen Harnisch. Industrial energy efficiency and climate change mitigation. *Energy efficiency*, 2(2):109, 2009.

[191] IETA. Cop23 summary report. Technical report, International Emissions Trading Association, November 2017.

[192] Luis Pérez-Lombard, José Ortiz, and Christine Pout. A review on buildings energy consumption information. *Energy and buildings*, 40(3):394–398, 2008.

[193] Vytautas Martinaitis, Edmundas Kazimieras Zavadskas, Violeta Motuzienė, and Tatjana Vilutienė. Importance of occupancy information when simulating energy demand of energy efficient house: A case study. *Energy and Buildings*, 101:64–75, 2015.

[194] Andrea Saltelli, Marco Ratto, Terry Andres, Francesca Campolongo, Jessica Cariboni, Debora Gatelli, Michaela Saisana, and Stefano Tarantola. *Global sensitivity analysis: the primer*. John Wiley & Sons, 2008.

[195] Tina Yu. Modeling occupancy behavior for energy efficiency and occupants comfort management in intelligent buildings. In *Machine Learning and Applications (ICMLA), 2010 Ninth International Conference on*, pages 726–731. IEEE, 2010.

[196] Ian Richardson, Murray Thomson, and David Infield. A high-resolution domestic building occupancy model for energy demand simulations. *Energy and buildings*, 40(8):1560–1566, 2008.

[197] Peter Stone and Manuela Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, 2000.

[198] Jacques Ferber. *Multi-agent systems: an introduction to distributed artificial intelligence*, volume 1. Addison-Wesley Reading, 1999.

[199] David Sturzenegger, Dimitrios Gyalistras, Vito Semeraro, Manfred Morari, and Roy S Smith. Brcm matlab toolbox: Model generation for model predictive building control. In *American Control Conference (ACC), 2014*, pages 1063–1069. IEEE, 2014.

[200] Gordon Prill. *Energy Estimating and Modeling Methods*.

[201] Norbert Harmati, Željko Jakšić, and Nikolai Vatin. Energy consumption modelling via heat balance method for energy performance of a building. *Procedia engineering*, 117:786–794, 2015.

[202] Han Chen, Paul Chou, Sastry Duri, Hui Lei, and Johnathan Reason. The design and implementation of a smart building control system. In *e-Business Engineering, 2009. ICEBE'09. IEEE International Conference on*, pages 255–262. IEEE, 2009.

[203] Bing Qiao, Kecheng Liu, and Chris Guy. A multi-agent system for building control. In *Intelligent Agent Technology, 2006. IAT'06. IEEE/WIC/ACM International Conference on*, pages 653–659. IEEE, 2006.

[204] Fernando Perez, Brian E Granger, and John D Hunter. Python: an ecosystem for scientific computing. *Computing in Science & Engineering*, 13(2):13–21, 2011.

[205] Fernando Pérez and Brian E Granger. Ipython: a system for interactive scientific computing. *Computing in Science & Engineering*, 9(3), 2007.

[206] Alexandros Nikitas, Ioannis Kougias, Elena Alyavina, and Eric Njoya Tchouamou. How can autonomous and connected vehicles, electromobility, brt, hyperloop, shared use mobility and mobility-as-a-service shape transport futures for the context of smart cities? *Urban Science*, 1(4):36, 2017.

[207] Han Chen, Paul Chou, Sastry Duri, Hui Lei, and Johnathan Reason. The design and implementation of a smart building control system. In *2009 IEEE International Conference on e-Business Engineering*, pages 255–262. IEEE, 2009.

[208] Paula Gori, Pier Luigi Parcu, and Maria Stasi. Smart cities and sharing economy. *Robert Schuman Centre for advanced studies research paper no. RSCAS*, 96, 2015.

[209] Anders Fremstad. Gains from sharing: sticky norms, endogenous preferences, and the economics of shareable goods. Technical report, Working Paper, 2014.

[210] Joyce Kim, Yuxun Zhou, Stefano Schiavon, Paul Raftery, and Gail Brager. Personal comfort models: predicting individuals' thermal preference using occupant heating and cooling behavior and machine learning. *Building and Environment*, 129:96–106, 2018.

[211] Bing Dong and Burton Andrews. Sensor-based occupancy behavioral pattern recognition for energy and comfort management in intelligent buildings. In *Proceedings of building simulation*, pages 1444–1451, 2009.

[212] Carlos Ernesto Ochoa and Isaac Guedi Capeluto. Evaluating visual comfort and performance of three natural lighting systems for deep office buildings in highly luminous climates. *Building and Environment*, 41(8):1128–1135, 2006.

[213] José Carlos R Alcantud and Annick Laruelle. To approve or not to approve: this is not the only question. 2012.

[214] William V Gehrlein and Peter C Fishburn. Condorcet's paradox and anonymous preference profiles. *Public Choice*, 26(1):1–18, 1976.

[215] TL Saaty. The analytic hierarchy process mcgraw-hill new york. *AGRICULTURAL ECONOMICS REVIEW*, 70, 1980.

[216] Hannu Nurmi. *Comparing voting systems*, volume 3. Springer Science & Business Media, 2012.

[217] Paulo Carreira, António Aguiar Costa, Vitor Mansur, and Artur Arsénio. Can hvac really learn from users? a simulation-based study on the effectiveness of voting for comfort and energy use optimization. *Sustainable cities and society*, 41:275–285, 2018.

[218] Ayşegül Uçar, Yakup Demir, and Cüneyt Güzeliş. A new facial expression recognition based on curvelet transform and online sequential extreme learning machine initialized with spherical clustering. *Neural Computing and Applications*, 27(1):131–142, 2016.

[219] Philipp Michel and Rana El Kaliouby. Real time facial expression recognition in video using support vector machines. In *Proceedings of the 5th international conference on Multimodal interfaces*, pages 258–264. ACM, 2003.

[220] Luca Mion, Ivan Pilati, Davi Macii, and Fabio Andreatta. Matching ontologies for emergency evacuation plans. Technical report, University of Trento, 2008.

[221] Jiyeong Lee and M-P Kwan. A combinatorial data model for representing topological relations among 3d geographical features in micro-spatial environments. *International Journal of Geographical Information Science*, 19(10):1039–1056, 2005.

[222] J Sime. Behavior in fire: panic or affiliation. *Department of Psychology, University of Surrey, UK*, 1984.

[223] Salman Azhar. Building information modeling (bim): Trends, benefits, risks, and challenges for the aec industry. *Leadership and management in engineering*, 11(3):241–252, 2011.

[224] Mehmet Yalcinkaya and Vishal Singh. Patterns and trends in building information modeling (bim) research: a latent semantic analysis. *Automation in Construction*, 59:68–80, 2015.

[225] Bilal Succar. Building information modelling framework: A research and delivery foundation for industry stakeholders. *Automation in construction*, 18(3):357–375, 2009.

[226] Gordon VR Holness. Bimgaining. 2008.

[227] Roozbeh Kangari and Tetsuji Yoshida. Automation in construction. *Robotics and Autonomous Systems*, 6(4):327–335, 1990.

[228] Madhav Prasad Nepal, Sheryl Staub-French, Rachel Pottinger, and Jiemin Zhang. Ontology-based feature modeling for construction information extraction from a building information model. *Journal of Computing in Civil Engineering*, 27(5):555–569, 2012.

[229] Han-Hsiang Wang, Frank Boukamp, and Tarek Elghamrawy. Ontology-based approach to context representation and reasoning for managing context-sensitive construction information. *Journal of Computing in Civil Engineering*, 25(5):331–346, 2010.

[230] TE El-Diraby and Hesham Osman. A domain ontology for construction concepts in urban infrastructure products. *Automation in Construction*, 20(8):1120–1132, 2011.

[231] Ferial Shayeganfar, Ardeshir Mahdavi, Georg Suter, Amin Anjomshoaa, A Zarli, and R Scherer. Implementation of an ifd library using semantic web technologies: A case study. *Proc., ECPPM 2008 eWork and eBusiness in Architecture, Engineering and Construction*, pages 539–544, 2008.

[232] Dave Beckett. Rdf/xml syntax specification (revised). w3c working draft, 2003, 2000.

[233] Thomas Liebich. Unveiling ifc2x4-the next generation of openbim. In *Proceedings of the 2010 CIB W78 Conference*, volume 8, 2010.

[234] Renaud Vanlande, Christophe Nicolle, and Christophe Cruz. Ifc and building lifecycle management. *Automation in construction*, 18(1):70–78, 2008.

[235] Pieter Pauwels and Walter Terkaj. Express to owl for construction industry: Towards a recommendable and usable ifcowl ontology. *Automation in Construction*, 63:100–133, 2016.