



## Article

# Smart Grid Ecosystem Modeling Using a Novel Framework for Heterogenous Agent Communities

Helder Pereira, Bruno Ribeiro, Luis Gomes  and Zita Vale \* 

Research Group on Intelligent Engineering and Computing for Advanced Innovation and Development (GECAD), Intelligent Systems Associated Laboratory (LASI), Polytechnic of Porto (P.PORTO), Rua Dr. António Bernardino de Almeida 431, 4200-072 Porto, Portugal

\* Correspondence: zav@isep.ipp.pt

**Abstract:** The modeling of smart grids using multi-agent systems is a common approach due to the ability to model complex and distributed systems using an agent-based solution. However, the use of a multi-agent system framework can limit the integration of new operation and management models, especially artificial intelligence algorithms. Therefore, this paper presents a study of available open-source multi-agent systems frameworks developed in Python, as it is a growing programming language and is largely used for data analytics and artificial intelligence models. As a consequence of the presented study, the authors proposed a novel open-source multi-agent system framework built for smart grid modeling, entitled Python-based framework for heterogeneous agent communities (PEAK). This framework enables the use of simulation environments but also allows real integration at pilot sites using a real-time clock. To demonstrate the capabilities of the PEAK framework, a novel agent ecosystem based on agent communities is shown and tested. This novel ecosystem, entitled Agent-based ecosystem for Smart Grid modeling (A4SG), takes full advantage of the PEAK framework and enables agent mobility, agent branching, and dynamic agent communities. An energy community of 20 prosumers, of which six have energy storage systems, that can share energy among them, using a peer-to-peer market, is used to test and validate the PEAK and A4SG solutions.

**Keywords:** multi-agent systems; smart grid modeling; agent mobility; agent branching



**Citation:** Pereira, H.; Ribeiro, B.; Gomes, L.; Vale, Z. Smart Grid Ecosystem Modeling Using a Novel Framework for Heterogenous Agent Communities. *Sustainability* **2022**, *14*, 15983. <https://doi.org/10.3390/su142315983>

Academic Editor: Mouloud Denai

Received: 25 October 2022

Accepted: 28 November 2022

Published: 30 November 2022

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

The proliferation of the smart grid context in power and energy systems is shifting how customers can use their energy, pay for the used energy, participate in the grid management, and the source of energy [1]. The sustainability of the grid and the massive usage of renewable generation sources are changing the energy grid and markets [2]. This demands the adoption of new information and communications technologies (ICTs) to address the issues raised by the need for smart grids [3]. Smart meters [4], energy management systems [5], and building energy management systems [5] are being tested and deployed almost everywhere, on a global scale.

The almost seamless integration of ICTs in smart grids enables a better digitalization of the grid, namely, on the customer side, also enabling the creation of a smart grid digital twin [6]. The use of near-real-time energy measurements and the ability to communicate bi-laterally with other smart grid players enable the appearance of new business models in smart grids, such as local energy markets [7], and active participation in demand response [8] and transactive energy [9], namely, peer-to-peer negotiation and energy transaction [10], contributing to the power and energy system's sustainability. However, due to their novelty and disruptive paradigm, the adoption of new business models requires the careful testing and validation of such models using proper simulators and emulators before going to pilot sites and real-world applications.

The operation and management of smart grids can be seen as a complex distributed system where multiple types of players interact with each other on multiple levels. These

levels can define microgrids, energy communities, aggregators, energy markets, or parts of the smart grids. Because of their complexity, management models are commonly supported by or even based on artificial intelligence (AI) [11], enabling the addressing of issues such as scalability [12], forecasting [13], uncertainty [14], and modeling [15].

The active participation of customers is one of the major changes in the power and energy system paradigm brought about by smart grids. In the literature, there is a significant number of participation models for energy customers [16–18]. Because it demands customer acceptance, it requires the engagement of customers. This can be promoted by the attribution of benefits or penalization in case of a lack of participation. However, the proposed models should be firstly heavily tested using a close-to-real environment to validate their results. To do so, the use of multi-agent systems (MASs) is commonly seen in the literature due to their distributed representation of players in complex systems.

The main motivation of this paper is the proposal of a novel open-source development framework for multi-agent systems able to work in simulated, emulated, and real environments that can be used in the several steps of the creation of new smart grid management models: conception, testing, validation, demonstration in pilots, and real deployment. This new MAS framework is named PEAK: Python-based framework for heterogeneous agent communities [19]. The proposed new framework was developed in Python, using as its core the Smart Python Agent Development Environment (SPADE). The novel framework goes beyond the other Python-based MAS frameworks because it allows the distributed execution of agents, the existence of multiple MASs, the integration of real loads, and a logging mechanism that helps the user to monitor and evaluate the performance of the agents and their MASs. The Python language was chosen because of its high number of packages and libraries for AI, enabling the developers of PEAK to have access to a variety of AI models and algorithms that they can easily integrate into their MASs.

To take proper advantage of PEAK's novel functionalities, this paper also presents a novel perspective of the use of MASs in smart grids, adding the functionalities of agent mobility and branching, allowing a complete distributed approach and an organic flow of agents to represent the dynamic contracts existing in a smart grid environment. To separate the novel development framework, PEAK, from the novel MAS perspective, the novel perspective for a multi-agent ecosystem able to better represent a smart grid and its players was named A4SG: Agent-based ecosystem for Smart Grid modeling.

To test and validate the technological functionalities of PEAK and the ability of A4SG to model a smart grid scenario, a case study was conceived and implemented considering 20 energy customers together with the existence of three energy forecast providers and a local market operator for peer-to-peer energy transactions. The agent communities of PEAK will represent contractual links in A4SG and will enable the representation of an energy community with 20 energy customers. The case study will present a complete scenario where energy customers are represented by intelligent agents that pursue the best-fitted energy forecast service provider to participate in peer-to-peer energy transactions. The case study will also provide a hybrid scenario where simulation and physical devices are synchronized to have realistic results. The integrated physical devices are six energy storage systems (ESS) that were attributed to six energy community members, enabling them to store and use energy from the ESS. The results show the ability of the proposed solutions to model smart grids and their constituents. The mobility of agents inside the modeled ecosystem is one of the main novelties of the proposed solutions and allows the representative agents of the energy community members to dynamically interact and establish connections to several service providers in order to maximize the member's benefits and minimize the energy costs.

After this first introductory section, the paper is divided into seven sections. Section 2 presents the state of the art regarding the application of MASs in smart grid modeling, while Section 3 focuses on the technological frameworks that enable the development of MASs, namely, frameworks developed in Python to facilitate the use of AI models and algorithms. PEAK is presented in Section 4, and A4SG is presented in Section 5. The case

study of an energy community with 20 energy customers that participate in peer-to-peer energy transactions is presented in Section 6. Section 7 shows the results of the case study, and Section 8 has the main conclusions of this work.

## 2. The Application of Multi-Agent Systems in Smart Grids

The general concept of an MAS relates to the implementation and development of several autonomous agents that interact with each other to achieve a common objective, despite the fact that each agent in the system may have its own particular objective(s) [20]. Generally, this type of system can be implemented in two ways. The system can utilize agents so that they cooperate and/or collaborate with one another in an effort to attain their goals, producing a solution for the system's ultimate objective [21]. On the other hand, the system can use agents in a competitive mode, in which they compete to accomplish individual goals; however, in this mode, it is more likely that some agents will fail to achieve their goals [21].

An MAS naturally has a decentralized architecture, where each agent performs some type of data processing or service execution and then shares information with other agents. Electric energy systems, with the evolution of the technologies used, and with the emergence of the smart grid paradigm, are increasingly being implemented and deployed in a decentralized way. Agents can also run in a distributed manner, across multiple physical and virtual locations, as seen on the smart grid, for instance, with distributed energy resources (DER). In addition, there are two topics in which MASs are very advantageous when applied to the smart grid [22]:

- The smart grid is a holistic system, and the failure of a component, such as a transformer or a transmission line, should not influence the entire activity and operation of the energy grid; thus, it must be fault tolerant [23]. Fault tolerance is enhanced in distributed management architectures such as MASs.
- As the smart grid is an ever-expanding system, it should be feasible to easily incorporate new resources and features using agents. Plug-and-play compatibility has been widely shown by MASs [24]. MASs may be scaled up by adding additional agents or by distributing them into a new environment with new resources and capabilities. Consequently, the ability of an MAS to be flexible and scalable is essential in the smart grid context.

In this way, the use of MASs for the representation, management, and control of the smart grid has gained a lot of interest, being a topic widely explored by the scientific community to address complex problems that the traditional approaches have not had the capacity to solve optimally [15].

In terms of more concrete applications of MASs to model smart grids, these can fall into a variety of areas, particularly due to the agents' ability to learn from their actions and simulate social behaviors [25], which are very useful tools for determining whether achieving the agent's objectives according to a set of requirements does not negatively impact other agents and the system in general. For instance, one of the most explored problems addressed using MASs is participation in demand response programs (DR) [25]. The use of MASs in DR can be applied for multiple settings, such as industrial contexts [26] and smart buildings [27].

Within the field of electrical energy, MASs have been demonstrating their usefulness in the representation of entities and their customers, such as microgrids [28], energy communities [29], and their management models [30]. In addition to representing entities belonging to the smart grid, MASs can also be used to individually represent customers and manage their resources, such as energy flexibility, and manage and support their active participation in the smart grid [31]. For instance, the authors of [32] proposed an MAS to simulate competitive electricity markets, the authors of [33] proposed an MAS to simulate DR participation, and the authors of [34] proposed an MAS to enable peer-to-peer energy trading in a microgrid. MASs can also be used to ensure stability in the grid by

enabling fault diagnosis and grid reconfiguration [35], voltage control [36], and voltage regulation [37].

With the developed study of the state-of-the-art of MASs, it is possible to conclude that these types of systems, particularly when applied to the representation and modeling of the smart grid, are becoming increasingly specialized and with specific objectives. Despite this being advantageous in some domains, this specialization of systems contributes to the problem associated with knowledge and service sharing between systems. In order to facilitate knowledge and service sharing, A4SG proposes several functionalities that make agents more intelligent in the management of knowledge and resources.

### 3. Frameworks for the Development and Implementation of Multi-Agent Systems

There are plenty of open-source frameworks and tools to help develop intelligent and autonomous agents as well as full autonomous MASs. It is possible to divide all frameworks and tools into two different categories based on the kind of system one wants to build. One category is the agent-based modeling frameworks, such as Mesa [38], NetLogo [39], and GridLAB-D [40], which help us build simulation environments, based on agents and their interactions, that represent real-world scenarios (e.g., objects, processes, and events). These frameworks are mostly used for research purposes because of their ability to facilitate the conception, integration, testing, and analysis of scientific models. The other category is the MAS frameworks, such as Java Agent DEvelopment (Jade) [41] and SPADE [42], which were designed to simplify the development and deployment of full autonomous MASs in real-world scenarios. Some of these frameworks also allow the creation of simulation environments.

This section will focus only on open-source MAS frameworks that had a regular update in the last five years. The selected frameworks are Jade, SPADE, Multiagent Development Kit (MaDKit) [43], Mava [44], and agentMET4FOF [45].

Jade is a framework written in Java that aims to build applications based on agents. It has an interactive interface for managing and monitoring the MAS as well as one interactive interface to deal with system debugging. This framework allows agent mobility and direct peer-to-peer communication using the Foundation for Intelligent Physical Agents (FIPA) standard specifications. Its documentation is complete, and there are some books related to this framework.

SPADE is a framework written in Python similar to the JADE framework. SPADE is built upon the Extensible Messaging and Presence Protocol (XMPP), which means that for the agents to communicate, an XMPP server must be used as the messages' broker. SPADE also allows FIPA's Agent Communication Language (ACL) and a web interface for each agent for individual managing and monitoring. Additionally, SPADE enables the easy development of agent-human interaction through XMPP. According to [46], SPADE is faster than JADE and has some official plugins to extend its base functionalities.

MaDKit is a Java framework for building distributed applications based on multiagent systems. One key concept of this framework is that the multiagent systems built using this framework are not agent-centered but specifically use the organizational architecture. In this architecture, each agent has a group and a role inside the community and does not rely on a broker agent for peer-to-peer communication. Additionally, this framework has interactive tools for constructing and visualizing simulation environments similar to what agent-based modeling frameworks have.

Mava is a Python framework designed for MASs specifically with the capacity of reinforcement learning. The goal is to facilitate the development of MASs that can work together to solve complex large problems while each agent has a decision-making component of a smaller part of the problem that is fueled by the reinforcement learning algorithms. In Mava, agents work in a cooperative manner.

The agentMET4FOF framework is focused on an MAS deployed in IoT environments. This framework targets industries and has a variety of tools for data processing and analysis.

This framework provides an interactive graphical interface for agent management and data analysis.

Table 1 shows the properties of each of the frameworks described earlier. These properties are the programming language they are written on, the date of the last update of the project main repository, the software licenses (if they provide any plugin that extends the base framework), the communication architecture that the agents are based on, if they comply with FIPA-ACL standard, and finally, systems the frameworks are targeted to develop.

**Table 1.** Framework comparison of MASs. The Last Update column was registered on the 3rd of August 2022.

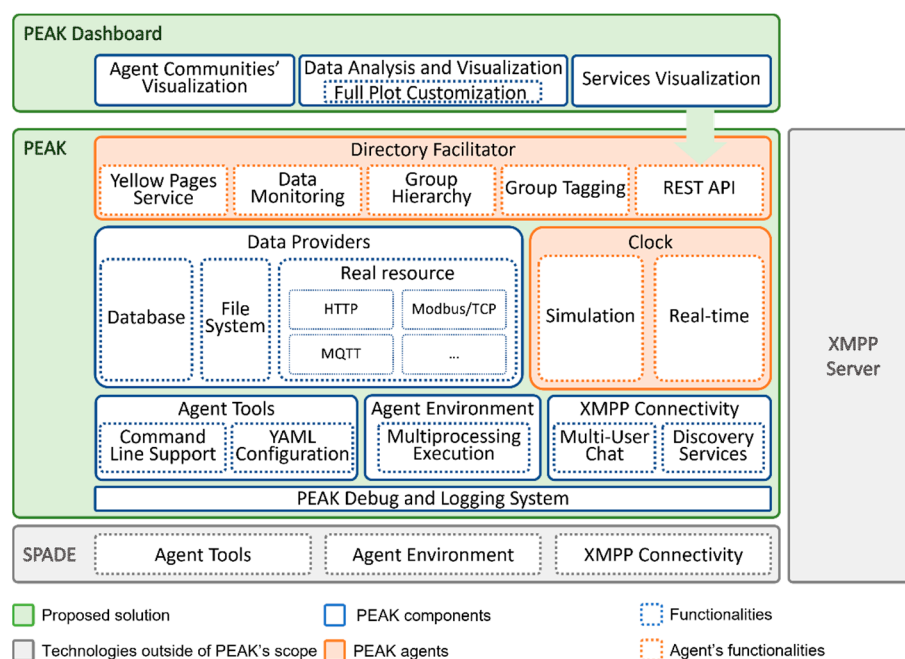
Framework	Language	Last Update	License	Plugins/ Modularity	Communication Architecture	FIPA-ACL	Targets
Jade	Java	8 June 2017	LGPL	Yes	Peer-to-peer	Yes	Real and Simulation Environments
SPADE	Python	1 August 2022	MIT	Yes	Client-Server	Yes	Real and Simulation Environments
MaDKit	Java	23 December 2021	CeCILL-C	No	Peer-to-peer	Yes	Real and Simulation Environments
Mava	Python	1 August 2022	Apache License 2	No	Peer-to-peer	No	Reinforcement Learning
agentMET4FOF	Peer-to Python peer	22 April 2022	LGPLv3	No	Peer-to-peer	No	IIoT

Every framework described in this section has its own benefits and usages, but when the focus is the development of an MAS of any kind and of any size with any application, the one that stands out is the SPADE framework. SPADE is a framework that is regularly updated, it is open source and has an MIT license. SPADE is a modular framework, allowing its expansion using plugins. It is FIPA complaint, and its communication architecture, despite being different from the mainstream architectures, ends up being more efficient because it provides many functionalities used for agent communication. This type of architecture allows human–agent interaction more easily, allows a more reliable communication between agents, because if one agent goes offline the conversations are not lost, and adds the ability to find the other agents more easily without the need of a facilitator agent. One advantage that SPADE has is the variety of libraries and packages that exist in Python and can be integrated into the agent, especially regarding the artificial intelligence domain. However, SPADE can be improved, as it lacks some useful functionalities, such as group chats between agents, a more robust and centralized web graphical interface to manage and monitor the MAS, a logging system, and a yellow page service.

#### 4. PEAK—A Novel Python-Based Framework for Heterogeneous Agent Communities

Python-based framework for heterogeneous agent communities (PEAK) is a developing framework that helps the user/developer to build and deploy multi-agent ecosystems. The architecture of PEAK is shown in Figure 1. The ecosystem is a prosperous environment for the MASs to share resources, socialize, and live together. PEAK permits the development of a wide range of MAS architectures. The MAS can be implemented in real-world scenarios, integrating IoT, for example, or used in simulated environments (e.g., smart grid simulation). Because there are a lot of frameworks for MAS development, PEAK was not built from the ground up. Instead, it uses SPADE as its core and adds functionalities. Similar to SPADE, PEAK is flexible and easily expandable. Beyond the additional functionalities, PEAK has an external Graphical User Interface (GUI), the Dashboard, for interactively monitoring the PEAK ecosystem.





**Figure 1.** The architecture of the PEAK framework.

SPADE is a multi-agent system framework that provides all of the necessary tools to develop and run any kind of MAS. To develop agents, SPADE offers different types of behaviors, for instance, periodic behaviors, cyclic behaviors, and one-time behaviors. The agents also have a knowledge system that allows them to store and search for knowledge. To communicate, SPADE's agents use a client-server architecture protocol named XMPP. This protocol is lightweight and fast and it is used for real-time chatting applications, making it suitable for MASs. XMPP provides the agents with contact lists and allows them to use the presence notification feature. A feature that allows an agent to notify its present state to its contact list (i.e., if it is online, away, or offline). However, SPADE lacks some useful functionalities, for instance, an internal clock to control simulation environments, easier integration with data providers, and a Directory Facilitator agent responsible for monitor and structuring the ecosystem around it. That is why PEAK was created, to empower an already useful framework, SPADE.

In SPADE, it is not possible to execute or configure the MAS through a command line interface (CLI). To facilitate that, PEAK adds a new CLI with the option to execute single agents, or multiple agents, and configure the execution options. To run multiple agents, the user needs a YAML configuration file describing the agents that he/she wants to run and the configurations for each one of them. Regarding the execution environment, SPADE uses multi-threading to execute them and, despite being fast, it does not conform to the definition of an agent described in the literature. An agent must be autonomous and independent, and a thread is always dependent on the other threads, either in terms of shared memory or in terms of concurrency. Therefore, PEAK added the option to run agents using multi-processing so users can choose the better way to approach their problem. The agents, in multi-processing, are faster and quicker to react because they are not dependent on each other. The downside of this is the increased memory usage, which is proportional to the number of agents running, and the slowness regarding the agents' start-up. Multi-processing would be better suited for problems that do not require too many agents. Multi-threading would be better if the problem requires a good amount of agents to run and if there is a lack of computation resources.

Regarding XMPP connectivity, SPADE only uses the basic functionalities for the agents to work and communicate. It uses peer-to-peer communication and the register and login mechanisms. To allow communities of agents, PEAK added two more XMPP functionalities

to the agents: the Multi-User Chat (MUC) and the Discovery Service (Disco). MUC allows the agents to create and join groups and exchange messages between them in a more efficient way; without it, the agents would have to manually send the messages to every single agent instead of one message to a group. However, with MUC, it is not possible to search for groups, and the agents are required to know the group name so they can join it. To overcome this, Disco comes into place. This functionality allows the agents to search for groups and other online agents present on the server.

PEAK's logging system allows the user to monitor and maintain the MAS. This system creates logging files for each agent that are saved and updated during the execution time. This functionality gives the user the ability to add logging information on the agents to better monitor and debug their behaviors.

In the literature, the MASs are often related to the use of a lot of data, either simulated data or data gathered from real devices. Because of that reason, PEAK facilitates the integration of these data providers with the MAS. Regarding the simulated data, PEAK has the possibility to integrate data from remote or local databases, as well as files from the file system (e.g., Excel, CSV). Regarding the real data, PEAK makes their integration possible by allowing the use of any communication protocols, for instance, HTTP/S, Modbus/TCP, MQTT, etc. In a simulated environment, the rate at which the agent reads the data is the same as the rate of PEAK's internal clock; otherwise, the user must configure the rate manually.

MASs have great utility when it comes to simulating environments, especially in the Smart Grid's research field. Therefore, to fulfill this need, PEAK created an internal clock capable of controlling the environment's time. The clock is controlled by an agent called Synchronizer and can be configured in three different ways. The first configuration, the default one, is the real-time clock. The user does not need any configuration, and agents are driven by the current time of the machine they are in. The second one is the simulation clock. This configuration allows the user to control the speed of the ticks. It can be faster or slower than the actual time, and the user can also configure how the time is perceived inside the simulation. That means the clock's ticks can either represent the current period (e.g., from 1 to 96) or can correspond to an actual date and time (e.g., 11 October 2000, 11:00). The last configuration is the dynamic clock. This configuration allows the user to change the time speed during the simulation. The time speed can be defined prior to the execution by the user or can be controlled dynamically by the agent. For example, this is useful in an energy market simulation where the user could fast forward periods of time without any activity.

PEAK has an agent called Directory Facilitator (DF) that has five main functionalities: yellow page service, data monitoring, group hierarchy, group tagging, and a REST API. The yellow page service is similar to the one found in the JADE framework. The idea is that the agents can register and search for services in the DF so that they can be shared between MASs. The data monitoring allows the MAS to export data generated during the execution to the DF. These data can then be used for further analysis or debugging of the MAS. The group hierarchy is an extension of the MUC functionality. It makes it possible for the MAS to build a hierarchy of groups that allows not only the organization and structuring of a community but also to send messages in a quicker and more efficient manner. Considering the structure to be a tree, the agents can echo a message through the branches of the hierarchical structure, from some node all the way down to the leaves. Another functionality related to the groups is group tagging. Group tagging helps the agents to identify and search for groups using tags instead of just using the name. Finally, the DF has an internal REST API that allows the user, with an external app, to obtain data from the PEAK ecosystem. This considers the data saved from the data monitoring, as well as the hierarchical structure of the groups and the services from the yellow page service. Therefore, the DF is an essential part of the framework because it allows the user to gather, monitor, and manage the information inside the ecosystem.

Last but not least, is the PEAK Dashboard, which can be used alongside the PEAK core. If the user does not have any app to take advantage of the DF, the PEAK Dashboard can fulfill that need. The PEAK Dashboard is a GUI that allows the user to visualize, interact, and monitor the PEAK ecosystem. The Dashboard must be initially connected to the DF REST API. One of the main features of the Dashboard is the ability to visualize the entire hierarchical architecture of the ecosystem and its agents' communications and interactions. The hierarchical architecture is represented in the form of a node graph in which a node represents a group. The node size is bigger or smaller according to the number of members the group has. The groups have also information related to the list of their members, their tags, and the level they are in the hierarchical structure. With this feature, the user can even filter the groups by their tags for easier navigation. Another feature is graph visualization. The Dashboard receives the MAS' data that the DF is monitoring and represents these data in the form of plots. The data are received in real-time by the DF and sent to the Dashboard. This makes the plots update dynamically in real-time throughout the execution of the MAS. The agents must explicitly create these plots in the DF, which are then sent to the Dashboard. Each agent can create the number of plots necessary, and they will all appear in the Dashboard in a grid-like manner. These plots, presented in Figure 2, are fully customizable by the final user, through the agent or in the Dashboard itself, and can be saved as images in the file system for external usage. The service visualization allows the user to see the services available in the DF's yellow page service. It also shows which agents are using each service and their historical usage.



**Figure 2.** PEAK Dashboard and data visualization example.

As described in this section, PEAK is a powerful framework for MASs. It allows the faster development and quick deployment of MASs. This is because PEAK removes a lot of boilerplates and facilitates the integration of every important aspect of an MAS. It also helps the user to debug and monitor PEAK's ecosystem for better and quicker maintenance. This framework is also good for research purposes because of its simulation features. All of this makes PEAK a very useful and practical framework for MAS development, especially in the energy research field, as shown in Section 5.

## 5. A4SG—A Novel Agent-Based Ecosystem for Smart Grid Modeling

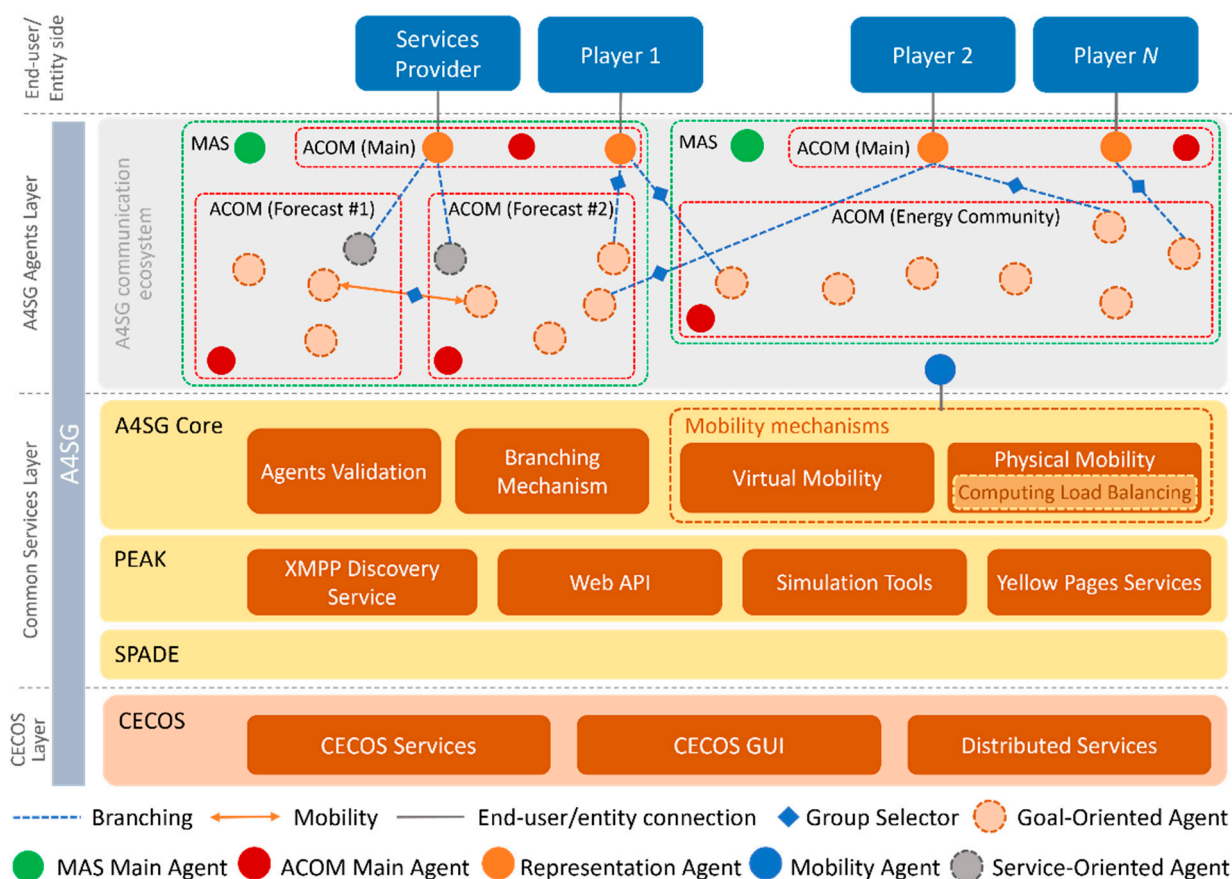
The evolution of the MAS, specifically in its application to the smart grid, has been continuous, with more and more novel approaches to handle complex challenges that centralized systems have numerous difficulties with. However, the fact is that most MASs are built for extremely specialized domains and settings, which makes it challenging to adapt them to new contexts. Furthermore, MASs are generally closed, and there is no contact with external systems, something that has been addressed, for instance, through the use of ontologies and semantics to standardize the representation of knowledge [47]. Thus, in order to counter the aforementioned problems, the need arises for a tool that allows the development of MASs that have a general application within the smart grid, especially with regard to their modeling and representation. In addition, it is important that different



MASs can communicate, interact, and even exchange agents so that there is an exchange of knowledge between the systems, which can facilitate their achievement of objectives.

The agent-based ecosystem for smart grid modeling (A4SG) is our proposal to address the problems and needs found in smart grid modeling, and it opens a new perspective when it comes to the application of MASs in smart grids. The A4SG is an ecosystem for smart grid modelling based on PEAK. The A4SG is used in this paper to test and validate peer-to-peer energy transactions models, in an energy community, powered by reinforcement learning.

A4SG employs PEAK groups to organize agents within the ecosystem. These groups represent MASs, as well as agent communities (ACOM), which, within the context of this ecosystem, are groups of agents with more particular objectives inside an MAS. In A4SG, seen in Figure 3, ACOMs can represent, for instance, a group of aggregators and its customers, an energy community, or a group to deploy a forecast service. There may be several levels in the hierarchy; firstly, there is always an MAS, and then there may be multiple levels of ACOM, which may be nested within one another; the lower the level in the hierarchy, the more specific the community's objective.



**Figure 3.** A4SG architecture with an example representing the developed case study in this paper.

The A4SG's architecture, represented in Figure 3, is divided into four main layers:

- **CECOS Layer:** the Citizen Energy Communities Operator System (CECOS), proposed in [48] and [49], is a web-based platform, initially conceived to manage citizen energy communities and their members. This platform was adapted to deal with different aggregation entities, and in the context of A4SG, it enables end-users to analyze the results of service participation, subscribe to services and aggregation entities, and manage the agents' knowledge base.
- **Common Services Layer:** this layer contains the services or mechanisms, provided by A4SG, PEAK, and SPADE, to all the agents registered in the ecosystem. SPADE

and PEAK were already described, so the focus here is on A4SG Core. The Agents Validation mechanism evaluates if a particular agent fits the criteria to participate in the ecosystem. For instance, it evaluates whether the agent's XMPP communication is functional, whether it is feasible to obtain and update the agent's data, and other needs that may be more relevant to the functionality that the agent wishes to execute. The branching, virtual mobility, and computing load-balancing mechanisms will be discussed in their respective subsections.

- **A4SG Agents Layer:** this layer represents the whole business logic of the ecosystem, namely, the MAS and ACOM hierarchy, and the registered agents. The Agents Representation Community (ARC) offers a centralized MAS for agents that want to participate in the ecosystem, without using their own machines to host them.
- **End-user/Entity Side:** layer to enable the representation of the end-users and entities registered in A4SG and to represent external agents (i.e., registered in A4SG, but deployed outside the hosts of the ecosystem).

There are several types of agents in A4SG, among which some are essential for the management of groups and the ecosystem in general, namely, (i) the MAS main agent, responsible for the management of the host where it is deployed, and for the sharing of information about the host (e.g., installed software, components status); (ii) ACOM main agent, that is very similar to the MAS main agent, but manages only one ACOM, which can be the representation, for instance, of an aggregator, retailer, or microgrid manager; (iii) Synchronizer agent, developed on top of PEAK's simulation tools, enables the execution of customizable simulations using XMPP's Multi-user Chat or PubSub protocols; and (iv) Mobility agent, which is responsible for both mobility processes in A4SG (i.e., virtual and physical mobilities), receiving the messages asking for mobility. On the other hand, there are agents that represent the users of the ecosystem. The main ones are the representation agents, which contain the entire knowledge base of the agent, which is essential in all of the agent's decision-making processes. These agents may have other agents that represent them in more specific objectives, called branch agents, more explored in the branching subsection.

In addition to what has been described and explored so far, A4SG offers an enhanced version of agent mobility and the novel concept of branching, which are both detailed in the following subsections.

### 5.1. Mobility

The existence of mobile agents in MASs is already an explored topic [50], although it is more common to exist in an MAS where the agents are static, that is, they do not carry out any type of mobility. However, mobility in agents can bring great advantages to agents and the entities they represent, the most evident being the ability that agents gain to adapt to their virtual and physical context. In A4SG, there are two types of mobility that agents have access to, which are intended to serve different purposes. It is important to note that messages requesting mobility must be sent to the mobility agent, which is unique in A4SG.

Firstly, there is virtual mobility, which is a less-explored topic. With the growth of service provider entities on the smart grid, it is important that energy customers have tools that allow them to adapt to their context and explore opportunities that can bring them greater advantages, both in terms of comfort and financially. Most of these opportunities represent the dynamic relationships present in the smart grid, and most of them work in the provider–client paradigm. Therefore, the main objective of this type of mobility is to allow agents to move between services and especially between aggregation entities, for instance, energy retailers and aggregators, according to the agent's evaluation of them. This evaluation can be completely customizable by the user so that the agent makes contextual decisions based on the preferences of the user it represents. In A4SG, this mobility is supported by the virtual mobility mechanism. In the context of this type of mobility, it is essential that the agent is located in the same physical host as the group to which the agent wants to move; if not, combined with virtual mobility, physical mobility comes into play.

Physical mobility consists of moving the execution of an agent to a different physical (e.g., laptop, server) or virtualized (e.g., docker running in a computer) host. This type of mobility is performed through weak mobilities, which consist of saving and restoring only the runtime values of the agent's properties, not including information, for instance, about the execution stack of the agent, which is considered in strong mobilities [51]. Physical mobility serves two different purposes: (i) it may not be convenient for a user to have his/her agent running in a specific host, for instance, in case the user is using his/her own local server and pretends to move the agent to a remote server in the cloud; or (ii) for purposes of balancing computational loads in the ecosystem, in which case it is the responsibility of the computing load balancing mechanism to manage the agent's mobility. The computing load balancing mechanism allows the distribution of the computational load across the hosts available in the A4SG ecosystem, ensuring that none of the machines are overloaded with agents, while others have low levels of computational effort.

Both types of mobility are supported by the group selector service, represented by blue squares in Figure 3. Agents can have customized ways of evaluating their mobility, but to ensure facilitation, this component allows the evaluation of general conditions, for example, in physical mobility, and more specifically in the computing load-balancing mechanism, it allows the evaluation of which machine is less overloaded to receive agents from machines that are most heavily loaded.

### 5.2. Branching

The notion of an agent is intimately tied to objectives since software agents are frequently created with the goal of achieving or searching for a certain objective. However, the reality is that, in multi-agent systems, agents are frequently overburdened with objectives, meaning that there are several focuses concurrently, making it difficult to complete objectives individually and generally. To solve this problem, the novel branching concept, supported by the branching mechanism and proposed by the authors, is available in A4SG to all agents. The branching mechanism prevents an agent from being overloaded by deploying new agents (i.e., branch agents) to handle new specific objectives. Consequently, a representation agent becomes a distributed entity, in which each branch agent represents it to achieve a single specific objective. In this manner, the ability to achieve objectives, individually and in a group, of a representation agent is boosted by splitting the objectives across several agents. Therefore, the branching technique enables agents to simultaneously engage in numerous aggregation entities and subscribe to many services. The branching, similar to the mobility, uses the group selector to recommend groups to the branch agents.

The two types of agents that can be created through branching are goal-oriented and service-oriented. Goal-oriented agents are designed to achieve a specific objective and maintain a direct link to their main agent (i.e., representation agent) to inform them of all decisions or participation in services. Participation in peer-to-peer markets and subscription to demand forecast services are examples of goal-oriented agents. Service-oriented agents, on the other hand, provide services to other agents. The service provided by the agent may be contained in its code or in an external application programming interface (API), with the service-oriented agent functioning as a gateway agent in this instance.

## 6. Case Study

In order to test PEAK and A4SG, a case study was used considering 20 prosumers that are members of the same energy community. Of those, six will have energy storage systems (ESS). Although the energy community members' energy profiles were created by using external data sources (i.e., [52,53]), the ESS are actual units available in the authors' research center, creating a hybrid simulation that will use simulated data and physical ESS. The six energy community members that have ESS, will be able to use them in a post-market phase to reduce costs related to the purchase and sale of purchases on the grid. The 20 members will be integrated into a group representing the energy community in the agent-based ecosystem of A4SG, with access to a peer-to-peer energy trading market.

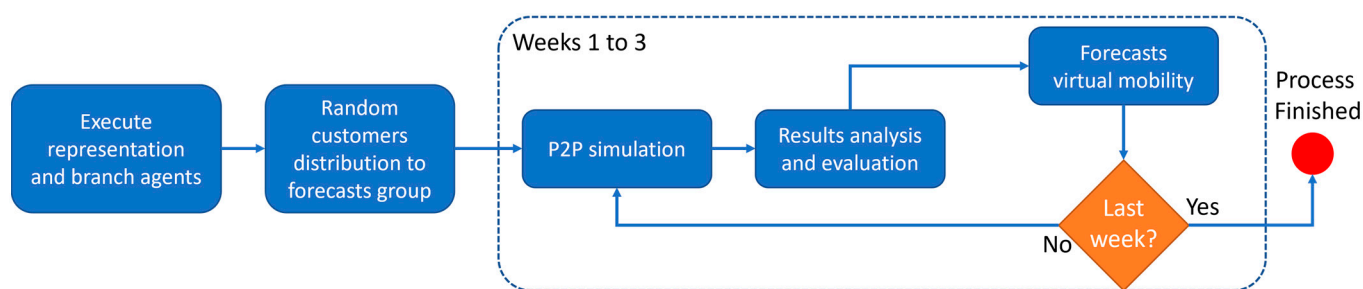
The considered energy community does not have an exclusive network for its members, thus being formed through contracts. Therefore, the network under consideration has consumers outside of the energy community, but energy transactions are only considered between the energy community members. The energy customer members are described in Table 2, where details about their consumption and generation profiles are explained. Taking advantage of the features provided by PEAK and A4SG, a three-week simulation will be carried out where customers can adapt to reduce the costs inherent to their energy resources. In this scenario, the agents are able to move between different forecast services, with different error profiles.

**Table 2.** Energy community members' description.

	Member 1	Member 2	Member 3	Member 4	Member 5	Member 6	Member 7	Member 8	Member 9	Member 10	Member 11	Member 12	Member 13	Member 14	Member 15	Member 16	Member 17	Member 18	Member 19	Member 20
Contracted Power (kVA)	5.75	6.9	3.45	6.9	3.45	3.45	6.9	10.35	5.75	6.9	2.3	3.45	4.6	4.6	6.9	2.3	3.45	4.6	6.9	13.80
Total Consumption (kW)	334	878	133	104	99	32	315	722	840	933	212	249	208	108	137	163	406	391	515	705
Total Generation (kW)	215	215	215	215	215	215	92	228	224	343	97	87	343	196	99	85	208	181	270	87
Generation Peak (kW)	1.6	1.6	1.6	1.6	1.6	1.6	1.2	2.6	2.6	3.8	1.2	1.0	3.6	2.2	1.2	1.0	2.4	2.0	3.0	1.0
Battery Capacity (kWh)	3.6	3.6	3.6	2.4	2.4	2.4	-	-	-	-	-	-	-	-	-	-	-	-	-	-

For members' participation in peer-to-peer energy transactions, the bids (i.e., energy amount to be transacted and maximum/minimum price accepted) are generated from pre-trained reinforcement learning policies. The algorithms used were the Deep Deterministic Policy Gradient (DDPG) and the twin delayed DDPG (TD3). DDPG and TD3 are two state-of-the-art reinforcement learning algorithms capable of dealing with continuous values for the observations (e.g., retailer energy prices, time of the day) and for the actions (i.e., amount of energy to trade and price to bid/ask). TD3 is an extended version of DDPG, covering some issues that the algorithm has, such as being very dependent on hyperparameter tuning. This type of intelligence shows the capabilities of PEAK, as well as A4SG, to integrate artificial intelligence models on top of agents with intelligent behaviors, creating an intelligent environment.

The case study execution responsibilities are divided between PEAK and A4SG. PEAK provides (i) the agents' development basis, (ii) the construction of the group's hierarchical tree, (iii) the simulation tools to accelerate the case study execution, and (iv) the connection to real ESS via a Modbus/TCP driver. On the other hand, A4SG's primary objectives are (i) to apply the branching mechanism to the members of the energy community so that they can create new agents to represent themselves in forecast services and peer-to-peer trading and (ii) to apply the mobility mechanism to branch agents so that agents can move between groups (i.e., in this case, to search for more suitable forecast services). Figure 4 shows the case study flowchart with the previously described steps.



**Figure 4.** Case study flowchart.

At the beginning of the simulation, different groups and agents are created, to guarantee the integrity of the ecosystem and the simulation itself, as well as to represent energy community members and service providers. The simulation has three forecast services and one market operator. The three forecasts are (i) a baseline forecast, using only mathematical models, (ii) a support vector machine (SVM) model, and (iii) an artificial neural network (ANN) model. Table 3 shows the groups and the agents that are part of each one of them. There are two MASs: (i) in ARC, there are the forecast groups, for the energy forecast service providers, represented by ACOMs, and (ii) in the Community MAS, there is the ACOM to represent the energy community and its members. Each MAS has a Main ACOM to host representation agents, while the other ACOMs contain branch agents. As seen in Section 5, an ACOM main agent is needed in each one of the ACOMs to manage the groups and to communicate information about the ACOM to other agents, such as the objective of the group and the events or service history. It is possible to conclude from the observation of Table 3, that all groups, with the exception of the forecast groups, maintain their agents from the beginning to the end of the simulation. In the case of forecast groups, the members are dynamic, as energy customers will move between them, taking into account the performed evaluation.

**Table 3.** Ecosystem structure to support the case study: groups and agents.

Group	Ecosystem Agents	Representation Agents	Branch Agents
ARC	1 MAS Main Agent 1 Mobility Agent	-	-
Main-ARC	1 ACOM Main Agent	20 Players 2 service providers	-
Forecast Baseline	1 ACOM Main Agent	-	1 Service-oriented (Forecast)
Forecast SVM	1 ACOM Main Agent	-	N goal-oriented (Players) in each group
Forecast ANN	1 ACOM Main Agent	-	
Community MAS	1 MAS Main Agent	-	-
Main-Community MAS	1 ACOM Main Agent	-	-
Energy Community with P2P	1 ACOM Main Agent	-	1 Service-oriented (Market Operator) 20 goal-oriented (Players)

*N* = dynamic (can change due to mobility).

Goal-oriented agents are programmed so that at the end of each week, they evaluate their context in the group in which they occur, and, if they are not satisfied, they can look for other options to improve their performance in the participating services. For forecast services, agents will check the mean absolute error (MAE) of their current forecast and can then compare it with that of other available forecasts to move to the service that provides them the lowest error.

To understand the benefits that the innovative characteristics of the ecosystem built by the combination of A4SG and PEAK provide to energy customers, two additional scenarios will be simulated to serve as a baseline to assess the full scenario with peer-to-peer market



and mobility between forecast services. The first scenario employs neither the peer-to-peer market nor the mobility proposed by A4SG, while the second scenario employs the peer-to-peer market without mobility.

## 7. Results and Discussion

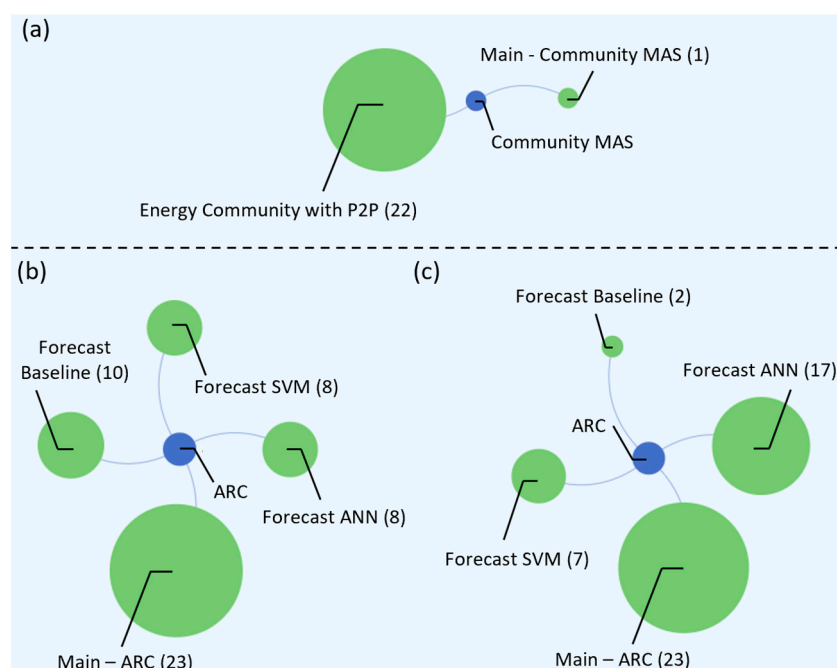
This section will be used exclusively to present the results related to the three-week simulation, especially regarding the advantages introduced by PEAK and A4SG for the energy community customers. The mobility carried out, the reduction of customers' energy costs, and the integration and use of real batteries will be analyzed. A comparison will be made with two other scenarios without the peer-to-peer market and the mobility mechanism.

The first situation to study is the mobility of agents across the forecast groups and to see the effect that this mobility has on the error associated with the energy forecast services. Table 4 shows the agents' distribution per forecast service provider group, where each agent's group in a specific week is marked by an "X". As can be seen in Table 4, the final distribution of the agents is quite different from the initial one, indicating that the agents were able to perform mobilities during the simulation to pursue better energy forecast results. At the end of the first week, the baseline forecast lost more than half of its customers, and by the end of the second week, it was even without any agents, proving that this was the worst service available to agents. Regarding the other two services, the one that uses an ANN as an algorithm is, in general, better than the service that uses an SVM. Even so, some of the agents stayed with the SVM forecast service provider until the last week, as it was the service that best adapted to their consumption and generation profiles. Overall, the community MAE dropped from 0.741 at the beginning of the simulation to 0.389 at the end of the simulation.

**Table 4.** Distribution of customers among forecast service providers during the simulation.

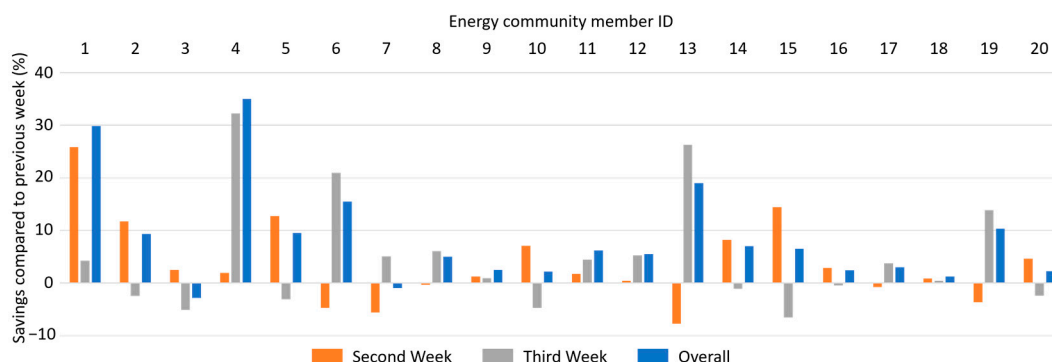
Customer ID	Week 1			Week 2			Week 3		
	Baseline	SVM	ANN	Baseline	SVM	ANN	Baseline	SVM	ANN
1	X			X					X
2			X			X			X
3	X				X				X
4	X					X			X
5		X			X			X	
6		X				X			X
7		X				X			X
8			X			X			X
9	X			X					X
10			X			X			X
11			X		X			X	
12	X			X					X
13			X			X			X
14			X			X			X
15	X				X			X	
16		X			X			X	
17	X					X			X
18	X					X			X
19		X				X		X	
20		X			X				X
Number of Customers	8	6	6	3	6	11	0	5	15
Average MAE	1.036	0.650	0.438	0.627	0.513	0.401	-	0.459	0.366

One of the great advantages provided by PEAK is the possibility of generating a chart with the groups that are part of the ecosystem so that it is possible to analyze them more graphically and intuitively. The chart from this case study is represented in Figure 5. Figure 5a shows the community MAS in which agents are static during the simulation, that is, the agents do not change groups during the simulation. On the other hand, Figure 5b,c, shows the ARC group, which hosts the forecast service provider groups and the agents move between them. Figure 5b shows the groups in the first week, and Figure 5c shows the same groups in the third week. As expected, the chart shows the mobilities that were processed in the ecosystem during the simulation, reflecting the information provided in Table 4.



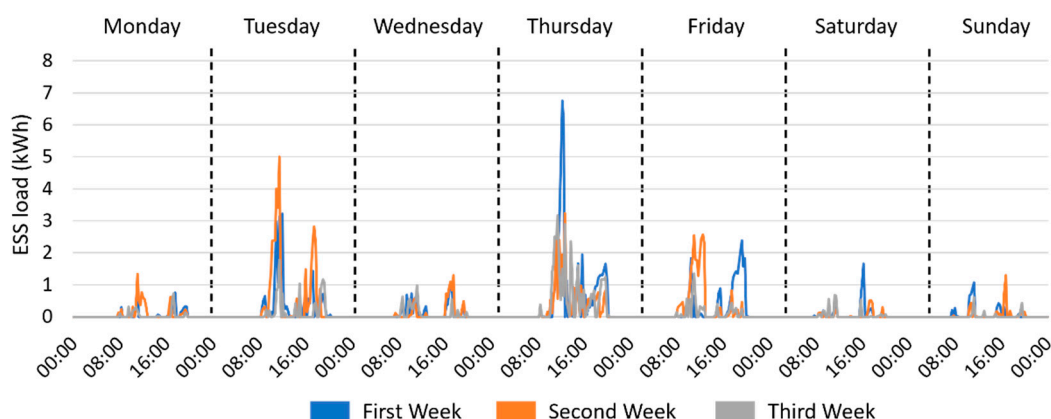
**Figure 5.** Ecosystem structure (extracted and adapted from the PEAK graphical interface) representing the names and the number of agents of agent communities: (a) agent communities representing the energy community, (b) agent communities in week 1 showing the distribution of agents among the forecast providers, and (c) agent communities in week 3 showing the distribution of agents among the forecast providers.

For these mobilities to have a positive effect on the energy customers, they must be used so that customers' participation in peer-to-peer markets is more effective and helps them to reduce energy costs. This analysis is represented in Figure 6. In this figure, the second and third weeks are compared for each player in terms of energy costs. As it is possible to observe, most of the customers benefited from the mobility due to the decrease in the energy forecast error. Only two of the energy community members were harmed (member 3 and member 7), that is, 90% of the members benefited from the change in their forecast service provider to a lower error forecast. Overall, comparing the third week of simulation with the first, there was a savings of 10.62 EUR in the community.



**Figure 6.** Energy costs in the second and third weeks compared to the first week of simulation.

Another factor that was influenced by the mobility of the agents is the use of ESS. Since ESS are used after the peer-to-peer market, to minimize the buying and selling of energy from and to the grid, the more accurate the bids for peer-to-peer, the lower the use of ESS. Figure 7 shows a comparison between weeks of the ESS' load for each period. As it is possible to observe, the ESS were less used with the advance of the weeks, and there was a 22% decrease in ESS' usage in the final week of the simulation compared to the initial week.



**Figure 7.** ESS' load comparison between weeks for each studied period.

To compare the three simulated scenarios, Table 5 shows the data regarding all scenarios. The greatest difference, as expected, is the introduction of the peer-to-peer market, especially with intelligent bid strategies provided by the reinforcement learning policies. Mobility allows for extra energy cost savings, proving that it can help energy customers to adapt to a context more advantageous to them, which in this case is the reduction of the energy forecasting model to be considered for the P2P market.

**Table 5.** Scenario comparison considering P2P and mobility.

Scenario	Community Overall Cost (EUR)	Mean Energy Price (EUR/kWh)
Without P2P and mobility	585.49	0.142
With P2P and without mobility	392.34	0.126
With P2P and mobility	377.85	0.119

Comparing this solution to other state-of-the-art approaches, it has the significant advantage of always being able to improve the participation by applying virtual mobility, allowing agents to move to other services that optimize the use of their energy resources. Comparing the proposed work in this paper with the ones proposed in [54,55], the initial

results (i.e., before mobility) are comparable, but when mobility is introduced, the outcomes of the proposed methodology in this paper differ in a positive way. Thus, customers are empowered in a dynamic manner in which they can determine the optimal approach to maximize their results, taking into account both the reduction of energy costs and the enhancement of comfort.

## 8. Conclusions

The use of multi-agent systems to model smart grids is a feasible solution that is supported by previous publications available in the literature. However, the existing multi-agent system frameworks have technical issues that impact the real test, simulation, and implementation of smart grid management and operation models.

When conceiving and developing sustainable energy-related models for smart grids, an important concept must be taken into account in artificial intelligence. The models available in artificial intelligence can provide important intelligent, fast, scalable, and knowledge-based tools that can significantly contribute to smart grids. Therefore, when choosing a multi-agent system framework, its connectivity and integration possibilities of artificial intelligence models must be studied.

This paper proposes a novel multi-agent system framework that was developed in Python to take proper advantage of the number of solutions available for data science and artificial intelligence. By using the PEAK framework, the developers will have at their disposal the vast set of libraries and packages that Python provides. The PEAK framework also provides, by default, a graphical user interface and the possibility to integrate emulated and real resources. The simulation motor of PEAK also provides the possibility to run the models in a simulation mode or in a real-time mode, allowing them to work in the same scenario with simulated, emulated, and real resources.

To demonstrate the real potential of the PEAK framework, the novel ecosystem of agent communities, A4SG, is proposed and demonstrated. In this ecosystem, the agents have free access to several communities where they can be integrated and moved around. The A4SG also proposes novel mechanisms regarding agent representation, agent mobility, and agent branching, that are not possible outside PEAK.

The case study demonstrates an operational energy community where its members can subscribe to energy forecast services in order to participate in peer-to-peer energy transactions. The demonstration was run using simulated and physical resources. The results showed that the model was successful and that the agent branching mechanism together with the agent mobility mechanism enabled an extra reduction of the price of energy in the energy community, by 9%.

As future work, both PEAK and A4SG can be addressed to increase their range of functionalities, opening doors to the integration of new business models in the smart grid and also in different domains. For PEAK, the idea would be to increase the performance of some tasks to make the execution of agents faster and lighter and create predefined templates for agents to speed up the development phase. The PEAK Dashboard should be more intuitive, and a search box and a pagination functionality in the plots section should be created for easier navigation. On the other hand, A4SG can be used to integrate new models on smart grids, for instance, to optimize participation in demand response or peer-to-peer trading to improve the customers' active participation dynamically, integrated with mobility and branching.

**Author Contributions:** Conceptualization, H.P., B.R., L.G. and Z.V.; methodology, H.P., B.R., L.G. and Z.V.; software, H.P. and B.R.; validation, H.P., B.R., L.G. and Z.V.; formal analysis, H.P., B.R. and L.G.; investigation, H.P. and B.R.; resources, Z.V.; data curation, H.P. and B.R.; writing—original draft preparation, H.P., B.R. and L.G.; writing—review and editing, H.P., B.R., L.G. and Z.V.; visualization, H.P. and B.R.; supervision, L.G. and Z.V.; project administration, L.G. and Z.V.; funding acquisition, Z.V. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work is supported by FEDER Funds through COMPETE program and by National Funds through FCT under projects UIDP/00760/2020, and UIDB/00760/2020.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Data available on request due to restrictions.

**Acknowledgments:** The authors acknowledge the work facilities and equipment provided by the GECAD research center (UIDB/00760/2020) to the project team.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

- Shaukat, N.; Ali, S.M.; Mehmood, C.A.; Khan, B.; Jawad, M.; Farid, U.; Ullah, Z.; Anwar, S.M.; Majid, M. A Survey on Consumers Empowerment, Communication Technologies, and Renewable Generation Penetration within Smart Grid. *Renew. Sustain. Energy Rev.* **2018**, *81*, 1453–1475. [\[CrossRef\]](#)
- Hache, E.; Palle, A. Renewable Energy Source Integration into Power Networks, Research Trends and Policy Implications: A Bibliometric and Research Actors Survey Analysis. *Energy Policy* **2019**, *124*, 23–35. [\[CrossRef\]](#)
- Dileep, G. A Survey on Smart Grid Technologies and Applications. *Renew. Energy* **2020**, *146*, 2589–2625. [\[CrossRef\]](#)
- Sun, Q.; Li, H.; Ma, Z.; Wang, C.; Campillo, J.; Zhang, Q.; Wallin, F.; Guo, J. A Comprehensive Review of Smart Energy Meters in Intelligent Energy Networks. *IEEE Int. Things J.* **2016**, *3*, 464–479. [\[CrossRef\]](#)
- Aliero, M.S.; Asif, M.; Ghani, I.; Pasha, M.F.; Jeong, S.R. Systematic Review Analysis on Smart Building: Challenges and Opportunities. *Sustainability* **2022**, *14*, 3009. [\[CrossRef\]](#)
- Jiang, Z.; Lv, H.; Li, Y.; Guo, Y. A Novel Application Architecture of Digital Twin in Smart Grid. *J. Ambient. Intell. Hum. Comput.* **2021**, *13*, 3819–3835. [\[CrossRef\]](#)
- Khorasany, M.; Mishra, Y.; Ledwich, G. Market Framework for Local Energy Trading: A Review of Potential Designs and Market Clearing Approaches. *IET Gener. Transm. Distrib.* **2018**, *12*, 5899–5908. [\[CrossRef\]](#)
- Honarmand, M.E.; Hosseini-zhad, V.; Hayes, B.; Shafie-khah, M.; Siano, P. An Overview of Demand Response: From Its Origins to the Smart Energy Community. *IEEE Access* **2021**, *9*, 96851–96876. [\[CrossRef\]](#)
- Zia, M.F.; Benbouzid, M.; Elbouchikhi, E.; Mueen, S.M.; Techato, K.; Guerrero, J.M. Microgrid Transactive Energy: Review, Architectures, Distributed Ledger Technologies, and Market Analysis. *IEEE Access* **2020**, *8*, 19410–19432. [\[CrossRef\]](#)
- Muhsen, H.; Allahham, A.; Al-halhouli, A.; Al-mahmodi, M.; Alkhraibat, A.; Hamdan, M. Business Model of Peer-to-Peer Energy Trading: A Review of Literature. *Sustainability* **2022**, *14*, 1616. [\[CrossRef\]](#)
- Bracco, S.; Rosales-Asensio, E.; González-Martínez, A.; Rosen, M.A.; Badidi, E. Edge AI and Blockchain for Smart Sustainable Cities: Promise and Potential. *Sustainability* **2022**, *14*, 7609. [\[CrossRef\]](#)
- Mohanty, A.; Samantaray, S.; Patra, S.S.; Ahmad, M.A.I.; Barik, R.K. An Efficient Resource Management Scheme for Smart Grid Using GBO Algorithm. In Proceedings of the 2021 International Conference on Emerging Smart Computing and Informatics, ESCI 2021, Pune, India, 5–7 March 2021.
- Abbasi, R.A.; Javaid, N.; Ghuman, M.N.J.; Khan, Z.A.; Ur Rehman, S. Amanullah Short Term Load Forecasting Using XGBoost. In *Workshops of the International Conference on Advanced Information Networking and Applications*; Springer: Cham, Switzerland, 2019; pp. 1120–1131.
- Kempitaya, T.; Sierla, S.; de Silva, D.; Yli-Ojanperä, M.; Alahakoon, D.; Vyatkin, V. An Artificial Intelligence Framework for Bidding Optimization with Uncertainty in Multiple Frequency Reserve Markets. *Appl. Energy* **2020**, *280*, 15918. [\[CrossRef\]](#)
- Mahela, O.P.; Khosravy, M.; Gupta, N.; Khan, B.; Alhelou, H.H.; Mahla, R.; Patel, N.; Siano, P. Comprehensive Overview of Multi-Agent Systems for Controlling Smart Grids. *CSEE J. Power Energy Syst.* **2022**, *8*, 115–131. [\[CrossRef\]](#)
- Lin, S.; Li, F.; Tian, E.; Fu, Y.; Li, D. Clustering Load Profiles for Demand Response Applications. *IEEE Trans. Smart Grid* **2019**, *10*, 1599–1607. [\[CrossRef\]](#)
- Muthirayan, D.; Baeyens, E.; Chakraborty, P.; Poolla, K.; Khargonekar, P.P. A Minimal Incentive-Based Demand Response Program with Self Reported Baseline Mechanism. *IEEE Trans. Smart Grid* **2020**, *11*, 2195–2207. [\[CrossRef\]](#)
- Arabzadeh, V.; Alimohammadisagvand, B.; Jokisalo, J.; Siren, K. A Novel Cost-Optimizing Demand Response Control for a Heat Pump Heated Residential Building. *Build. Simul.* **2018**, *11*, 533–547. [\[CrossRef\]](#)
- Ribeiro, B.; Pereira, H.; Gomes, L.; Vale, Z. Python-Based Ecosystem for Agent Communities Simulation. In *International Workshop on Soft Computing Models in Industrial and Environmental Applications*; Springer: Cham, Switzerland, 2023; pp. 62–71. [\[CrossRef\]](#)
- Dorri, A.; Kanhere, S.S.; Jurdak, R. Multi-Agent Systems: A Survey. *IEEE Access* **2018**, *6*, 28573–28593. [\[CrossRef\]](#)
- Amirkhani, A.; Barshooi, A.H. Consensus in Multi-Agent Systems: A Review. *Artif. Intell. Rev.* **2022**, *55*, 3897–3935. [\[CrossRef\]](#)
- Nair, A.S.; Hossen, T.; Champion, M.; Selvaraj, D.F.; Goveas, N.; Kaabouch, N.; Ranganathan, P. Multi-Agent Systems for Resource Allocation and Scheduling in a Smart Grid. *Technol. Econ. Smart Grids Sustain. Energy* **2018**, *3*, 15. [\[CrossRef\]](#)
- Mohammadali, A.; Haghighi, M.S. A Privacy-Preserving Homomorphic Scheme with Multiple Dimensions and Fault Tolerance for Metering Data Aggregation in Smart Grid. *IEEE Trans. Smart Grid* **2021**, *12*, 5212–5220. [\[CrossRef\]](#)



24. Alseyat, A.; Ullah, M.H.; Park, J.D. Multi-Agent System-Based Plug-and-Play Energy Management System for DC Microgrids. In Proceedings of the 2020 IEEE 9th Power India International Conference (PIICON), Tempe, AZ, USA, 11–13 April 2021. [\[CrossRef\]](#)
25. González-Briones, A.; De La Prieta, F.; Mohamad, M.; Omatu, S.; Corchado, J. Multi-Agent Systems Applications in Energy Optimization Problems: A State-of-the-Art Review. *Energies* **2018**, *11*, 1928. [\[CrossRef\]](#)
26. Woltmann, S.; Kittel, J. Development and Implementation of Multi-Agent Systems for Demand Response Aggregators in an Industrial Context. *Appl. Energy* **2022**, *314*, 118841. [\[CrossRef\]](#)
27. Kem, O.; Ksontini, F. A Multi-Agent Approach to Energy Optimisation for Demand-Response Ready Buildings. In *Artificial Intelligence Techniques for a Scalable Energy Transition: Advanced Methods, Digital Technologies, Decision Support Tools, and Applications*; Springer: Cham, Switzerland, 2020; pp. 77–107. [\[CrossRef\]](#)
28. Lee, J.-W.; Kim, M.-K.; Kim, H.-J. A multi-agent based optimization model for microgrid operation with hybrid method using game theory strategy. *Energies* **2021**, *14*, 603. [\[CrossRef\]](#)
29. Reis, I.F.G.; Gonçalves, I.; Lopes, M.A.R.; Antunes, C.H. A Multi-Agent System Approach to Exploit Demand-Side Flexibility in an Energy Community. *Util. Policy* **2020**, *67*, 101114. [\[CrossRef\]](#)
30. Azeroual, M.; Lamhamdi, T.; El Moussaoui, H.; El Markhi, H. Simulation Tools for a Smart Grid and Energy Management for Microgrid with Wind Power Using Multi-Agent System. *Wind Eng.* **2020**, *44*, 661–672. [\[CrossRef\]](#)
31. Coelho, V.N.; Weiss Cohen, M.; Coelho, I.M.; Liu, N.; Guimarães, F.G. Multi-Agent Systems Applied for Energy Systems Integration: State-of-the-Art Applications and Trends in Microgrids. *Appl. Energy* **2017**, *187*, 820–832. [\[CrossRef\]](#)
32. Wu, J.; Wang, J.; Kong, X. Strategic Bidding in a Competitive Electricity Market: An Intelligent Method Using Multi-Agent Transfer Learning Based on Reinforcement Learning. *Energy* **2022**, *256*, 124657. [\[CrossRef\]](#)
33. Golmohamadi, H.; Keypour, R.; Bak-Jensen, B.; Pillai, J.R. A Multi-Agent Based Optimization of Residential and Industrial Demand Response Aggregators. *Int. J. Electr. Power Energy Syst.* **2019**, *107*, 472–485. [\[CrossRef\]](#)
34. Gomes, L.; Vale, Z.; Corchado, J.M. Multi-Agent Microgrid Management System for Single-Board Computers: A Case Study on Peer-to-Peer Energy Trading. *IEEE Access* **2020**, *8*, 64169–64183. [\[CrossRef\]](#)
35. Rahman, M.S.; Isherwood, N.; Oo, A.M.T. Multi-Agent Based Coordinated Protection Systems for Distribution Feeder Fault Diagnosis and Reconfiguration. *Int. J. Electr. Power Energy Syst.* **2018**, *97*, 106–119. [\[CrossRef\]](#)
36. Wang, S.; Duan, J.; Shi, D.; Xu, C.; Li, H.; Diao, R.; Wang, Z. A Data-Driven Multi-Agent Autonomous Voltage Control Framework Using Deep Reinforcement Learning. *IEEE Trans. Power Syst.* **2020**, *35*, 4644–4654. [\[CrossRef\]](#)
37. Wang, X.; Wang, C.; Xu, T.; Guo, L.; Li, P.; Yu, L.; Meng, H. Optimal Voltage Regulation for Distribution Networks with Multi-Microgrids. *Appl. Energy* **2018**, *210*, 1027–1036. [\[CrossRef\]](#)
38. Kazil, J.; Masad, D.; Crooks, A. Utilizing Python for Agent-Based Modeling: The Mesa Framework. In *Proceedings of the Social, Cultural, and Behavioral Modeling*; Thomson, R., Bisgin, H., Dancy, C., Hyder, A., Hussain, M., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 308–317.
39. Wilensky, U. *NetLogo*; Northwestern University: Evanston, IL, USA, 1999.
40. Chassin, D.P.; Fuller, J.C.; Djilali, N. GridLAB-D: An Agent-Based Simulation Framework for Smart Grids. *J. Appl. Math.* **2014**, *2014*, 492320. [\[CrossRef\]](#)
41. Bellifemine, F.; Poggi, A.; Rimassa, G. *JADE—A FIPA-Compliant Agent Framework*; The Practical Application Company Ltd.: London, UK, 1999; pp. 97–108. (In English)
42. Palanca, J.; Terrasa, A.; Julián, V.; Carrascosa, C. SPADE 3: Supporting the New Generation of Multi-Agent Systems. *IEEE Access* **2020**, *8*, 182537–182549. [\[CrossRef\]](#)
43. Gutknecht, O.; Ferber, J. The MadKit Agent Platform Architecture. In *Workshop on Infrastructure for Scalable Multi-Agent Systems at the International Conference on Autonomous Agents*; Springer: Berlin/Heidelberg, Germany, 2000; Volume 1887.
44. Pretorius, A.; Tessler, K.; Smit, A.P.; Formanek, C.; Grimbly, S.J.; Eloff, K.; Danisa, S.; Francis, L.; Shock, J.; Kamper, H.; et al. Mava: A Research Framework for Distributed Multi-Agent Reinforcement Learning. *arXiv* **2021**, arXiv:2107.01460.
45. Ludwig, B.; Bang, Y.; Prasad, A.; Lulic, H.; Gruber, M.; Kok, G. Met4FoF/agentMET4FOF: v0.13.2 (v0.13.2). Zenodo. 2022. Available online: <https://zenodo.org/record/5965562#.Y4bc1hVBxPY> (accessed on 24 October 2022).
46. Radhakrishnan, G.; Chithambaram, V.; Shunmuganathan, K.L. Comparative Study of Jade and Spade Multi Agent System. *Int. J. Adv. Res.* **2018**, *6*, 1035–1042. [\[CrossRef\]](#) [\[PubMed\]](#)
47. Santos, G.; Pinto, T.; Vale, Z. Ontologies to Enable Interoperability of Multi-Agent Electricity Markets Simulation and Decision Support. *Electronics* **2021**, *10*, 1270. [\[CrossRef\]](#)
48. Pereira, H.; Gomes, L.; Faria, P.; Vale, Z.; Coelho, C. Web-Based Platform for the Management of Citizen Energy Communities and Their Members. *Energy Inform.* **2021**, *4*, 43. [\[CrossRef\]](#)
49. Pereira, H.; Ribeiro, B.; Gomes, L.; Vale, Z. CECOS: A Centralized Management Platform Supported by Distributed Services to Represent and Manage Resources Aggregation Entities and Its End-Users in a Smart Grid Context. In *IFAC-PapersOnLine*; Elsevier: Amsterdam, The Netherlands, 2022; Volume 55, pp. 309–314.
50. Mazdin, P.; Rinner, B. Coordination of Mobile Agents for Simultaneous Coverage. In Proceedings of the PRIMA 2019: Principles and Practice of Multi-Agent Systems: 22nd International Conference, Turin, Italy, 28–31 October 2019; Springer: Cham, Switzerland, 2019. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Volume 11873, pp. 170–185. [\[CrossRef\]](#)

51. Mitrovic, D.; Ivanovic, M.; Budimac, Z.; Vidakovic, M. An Overview of Agent Mobility in Heterogeneous Environments. In Proceedings of the WASA, Chengdu, China, 11–13 August 2011; pp. 52–58.
52. Goncalves, C.; Barreto, R.; Faria, P.; Gomes, L.; Vale, Z. Dataset of an Energy Community's Consumption and Generation with Appliance Allocation for One Year. *Data Brief* **2022**, *45*, 108590. [CrossRef]
53. Working Group on Intelligent Data Mining and Analysis (IDMA) Open Data Sets. Available online: <https://site.ieee.org/pes-iss/data-sets/> (accessed on 1 March 2022).
54. Xu, S.; Zhao, Y.; Li, Y.; Zhou, Y. An Iterative Uniform-Price Auction Mechanism for Peer-to-Peer Energy Trading in a Community Microgrid. *Appl. Energy* **2021**, *298*, 117088. [CrossRef]
55. Zheng, B.; Wei, W.; Chen, Y.; Wu, Q.; Mei, S. A Peer-to-Peer Energy Trading Market Embedded with Residential Shared Energy Storage Units. *Appl. Energy* **2022**, *308*, 118400. [CrossRef]