Internet of Things 37 (2026) 101906

Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.elsevier.com/locate/iot

Home automation interoperability: two decades of lessons learned and future prospects into the development of IoT ecosystems

Dario Russo [a], Vittorio Miori [a], Gabriele Tolomei [a,b], Dimitri Belli [a,*]

[a] Institute of Information Science and Technologies, National Research Council (CNR-ISTI), Via G. Moruzzi 1, Pisa, 56124, Tuscany, Italy
[b] La Sapienza, University of Rome, Via Salaria 113, Rome, 00199, Lazio, Italy

ARTICLE INFO

Keywords:
Home automation
Interoperability frameworks
Taxonomy
IoT Middleware
Integration platforms
Smart home systems

ABSTRACT

Home Automation (HA) is one of the main applications of the Internet of Things (IoT), with the aim of improving users' quality of life within their living spaces. Technologies are growing rapidly, moving from simple control of isolated home devices to global integration with worldwide services. With the entry of large companies such as Google, Amazon and Apple into the HA market, interoperability between different automation ecosystems has become a central challenge for research. This study defines the concept of interoperability in this context and proposes a taxonomy based on years of experience to illustrate a method for a comparative study of frameworks designed to achieve interoperability between different systems. As a result, the presented taxonomy establishes a unified language and a set of criteria for evaluating integration strategies, improving understanding of architectural choices and decision-making processes in the selection or development of solutions for heterogeneous smart home systems.

1. Introduction

Today, advances in electronic and communication technologies [1] have allowed wide-ranging exploitation of powerful miniaturized computers, sensors, actuators, and efficient computer networks. These technologies have the potential to develop further and thereby bring to market ever more sophisticated and efficient devices at increasingly competitive prices.

The Internet of Things (IoT) [2,3], a key component of the current and future Internet, refers to a dynamic global network infrastructure characterized by self-configuring capabilities based on standardized communication protocols. In the Internet of Things (IoT), devices have distinct identities and combine physical properties with virtual attributes, communicating through intelligent interfaces and integrating seamlessly into networked ecosystems [4]. The IoT aims to enable novel interactions between devices and users, as well as between devices themselves, by exploiting the ubiquitous presence of various smart objects such as Radio-Frequency Identification (RFID) tags, sensors, actuators, and mobile devices [5]. These interconnected entities collaborate and communicate through unified protocols and unique addressing schemes to achieve common goals [6]. Home Automation (HA), instead, refers to technologies that enable the automation and integration of household systems and activities through intelligent devices and systems [7]. It enables the automated control of lighting, HVAC (i.e., Heating, Ventilation and Air Conditioning), appliances, gates, door locks and other subsystems based on specific objectives. HA systems reduce operational costs and enhance energy efficiency, user comfort, and safety and security while accommodating individual preferences. For elderly and disabled individuals, these systems promote

* Corresponding author.
E-mail address: dimitri.belli@isti.cnr.it (D. Belli).

https://doi.org/10.1016/j.iot.2026.101906
Received 14 August 2025; Received in revised form 8 December 2025; Accepted 16 February 2026
Available online 18 February 2026

autonomy and reduce dependence on caregivers and institutional care [8]. HA adoption has increased in recent years, driven by greater affordability and the widespread availability of user-friendly interfaces accessible via smartphones and tablets.

While the IoT encompasses a network that connects people and devices anytime, anywhere, HA represents a human-centric application of IoT that focuses on improving the quality of life for users [9].

HA is a key application domains within the IoT landscape. The underlying technologies are rapidly evolving, transitioning from isolated device control towards comprehensive integration of residential environments with global digital services. Technologies such as KNX [10], UPnP [11], and LonWorks [7], are already widely deployed in modern homes, enabling access to a broad and expanding set of applications and services. These protocols continue to attract significant attention in the research community [12]. Moreover, significant new players have entered the HA market in recent years, including Google with Google Home, Amazon with Amazon Echo, and Apple with Apple Home Kit [13]. The integration of different HA ecosystems into the global IoT is therefore a specific research challenge, and the main objective of scientific research in this area is to find solutions to interoperability issues that are in line with open and widely recognized standards.

A challenge currently facing the HA research community is the lack of standardization in communication protocols and system architectures. The HA market remains highly fragmented, with vendors often pursuing proprietary solutions aimed at securing exclusive customer relationships. These strategies can create market entry barriers for new manufacturers and limit interoperability by encouraging consumer dependence or specific ecosystems. It follows that users are frequently restricted to devices compatible only with a particular vendor's protocol. Addressing this issue requires the development of interoperable computing frameworks capable of abstracting the distinct characteristics of diverse technologies, thereby enabling seamless coexistence within unified HA environments.

Recently, various consortia and alliances have introduced new middleware solutions. These solutions promise to enable interoperability among heterogeneous HA ecosystems. Consequently, any existing or future HA technology has the potential to be interoperable. A case point is the Connectivity Standard Alliance, formerly known as Zigbee Alliance, which recently published the Specification 1.0 of Matter[1], a royalty-free, open-source, connectivity standard which is intended to enable cross-platform interoperability between smart devices, mobile apps, and cloud services. The Matter specification outlines the basic requirements for an interoperable application layer solution that enables smart home devices produced by different vendors to communicate with each other over the version six of the Internet Protocol (IPv6) [14].

A substantial proportion of the existing literature on HA has primarily focused on the analysis of interoperability standards and protocols. However, considerably less attention has been devoted to the practical implementation and comparative evaluation of HA frameworks that embody these standards. As a result, the current body of research provides only a partial understanding of how interoperability is achieved in operational systems and the limitations that persist in real deployments.

The current work aims to address this gap by providing a systematic, comparative analysis of major HA frameworks, examining their ability to support interoperability across heterogeneous devices and technologies through a unified analytical approach. Furthermore, a novel interoperability taxonomy is introduced, specifically tailored to HA systems. This taxonomy offers a structured perspective to assess existing solutions and guide future development, as it was not previously formalized in the literature.

Accordingly, our main contributions are as follows:

- **A domain-specific definition of interoperability in HA**, clearly distinguished from broader IoT interoperability.
- **A new taxonomy for HA interoperability**, serving as a conceptual tool to categorize interactions, layers, and mechanisms across frameworks.
- **A comparative evaluation of prominent HA frameworks**, using the proposed taxonomy to expose their interoperability strengths, limitations, and persistent challenges.

Together, these contributions offer a concise yet thorough overview of the field and lay the groundwork for future advancements in interoperable HA ecosystems.

The remainder of this paper is structured as follows. Section II reviews related work. Section III defines interoperability in the IoT context, with a specific focus on its distinctions within HA. Section IV presents a taxonomy tailored to HA interoperability. Section V applies this taxonomy to compare a representative set of prominent frameworks. Finally, Section VI discusses the key findings and outlines future research directions.

2. Related work

Since the late 1990s, the scientific community has worked to summarize and harmonize the concept and levels of interoperability. Although the extensive literature on the topic contains many proposed approaches, the multitude of technical areas involved and the large number of systems referenced hinder efforts to establish a universally recognized, comprehensive definition of interoperability.

The U.S. Department of Defense proposed the Levels of Information Systems Interoperability (LISI) model, which focuses on the technical aspects and complexity of system interoperability. It classifies interoperability among computer systems into five levels based on the communication environment under consideration [15]. However, the LISI model does not address the environmental and organizational aspects that contribute to the construction and maintenance of interoperable systems. Instead, these facets are covered

[1] https://csa-iot.org/all-solutions/matter/

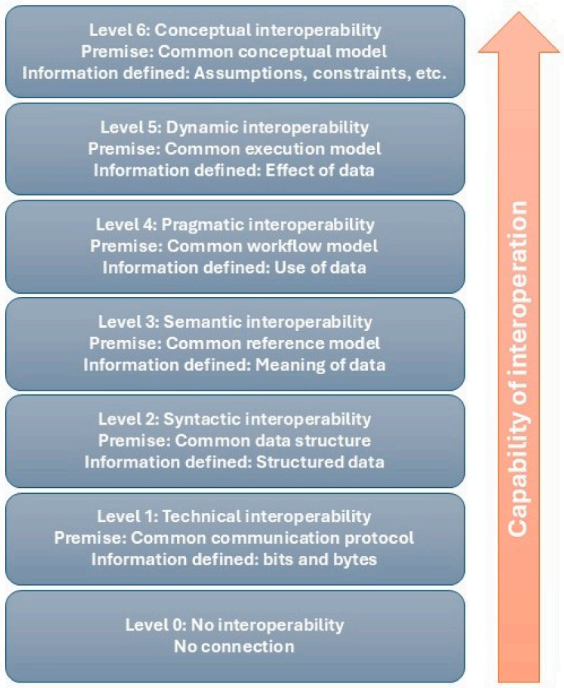

Fig. 1. Conceptual representation of home automation interoperability.

by the Organizational Interoperability Maturity Model [16], which extends the LISI model in terms of organizational structures, policies, and processes, offering a more holistic view of interoperability.

Tolk and Muguira [17] subsequently developed the Levels of Conceptual Interoperability Model (LCIM), which addresses conceptual levels of interoperability that go beyond technical models such as LISI, tackling the challenge from a more abstract and semantic perspective. Tolk also introduced the Model-based Data Engineering (MDBE) approach [18], an engineering method designed to achieve interoperability within the context of simulation theory. This approach bridges conceptual and technical system design and has broader applicability beyond simulation. The LCIM has undergone several refinements over the past decades, with its most recent version proposed by Turnitsa [19]. The levels of interoperability defined in this updated model are shown in Fig. 1.

The LCIM illustrates that achieving composability between services requires addressing interoperability at multiple conceptual levels, from technical connectivity to shared semantics.

In Ide and Pustejovsky [20], the authors define interoperability as a *measure of the degree to which diverse systems, organizations, and/or individuals can work together to achieve a common goal*. In the context of computer systems, particularly in language processing, the authors argue that the focus is increasingly shifting from syntactic to semantic interoperability. Syntactic interoperability relies on well-defined data formats and communication protocols to enable data exchange. However, while systems may be able to process this data, there is no guarantee that each interpretation will be the same.

On the other hand, semantic interoperability exists when systems can automatically interpret exchanged information meaningfully and accurately to produce useful results according to a common information exchange reference model. Semantic interoperability implies that the meaning of any information sent by one system will be the same as the one understood by the receiver system.

A common approach to achieving semantic interoperability is through ontologies, which provide an explicit specification of a shared conceptualization. Although a universal ontology would be ideal, it is considered impractical because new concepts constantly emerge and existing terms evolve in meaning [21]. A more feasible approach is to use a foundation (or upper) ontology that defines a limited set of primitive concepts from which domain-specific concepts can be derived. This approach could enable consistent interpretation across systems. However, no foundation ontology has achieved widespread adoption yet, so it remains a challenge for the future.

Although conceptual models such as ontologies theoretically support semantic alignment, there is still a lack of practical, structured tools for evaluating interoperability in specific domains, such as HA. Taxonomies can fill this gap by organizing domain knowledge in a way that supports analysis and comparison. Depending on the specific aspect addressed, the literature on HA has proposed various taxonomies.

For instance, Taiwo et al. [22] define a taxonomy of IoT-based smart HA systems, related technologies, future directions, and challenges, discussing system design approaches, strengths, and weaknesses. Andraschko et al. [23] define a taxonomy to identify the main objectives that smart home technology must address, with the intent of enabling researchers and organizations to better design

and evaluate the impacts of these technologies. De Franco et al. [24] focus on smart home research themes, proposing an analysis and a taxonomy of these concepts.

Fakhr Hosseini et al. [25] propose a taxonomy for HA aimed at establishing universal definitions within a fragmented context, where inconsistent terminology often increases confusion. By engaging domain experts through online surveys and interviews, the authors developed a new taxonomy that standardizes terminology, facilitates communication among researchers, designers, and developers, and enhances consumer awareness. Mzili et al. [26] present an insightful study on interoperability in the IoT domain, reviewing existing taxonomies and identifying the main barriers that hinder system interoperability, with particular attention to the healthcare domain. Ezugwu et al. [27] provide an extensive overview of emerging trends and future directions in smart home technologies. In their work, the authors propose a taxonomy that categorizes the main technological components and challenges of smart home ecosystems, among which interoperability remains a key issue yet to be fully addressed.

Collectively, these prior works contribute meaningful taxonomic perspectives on smart home and IoT systems. Taiwo et al. [22] classify IoT-based HA technologies and design approaches; Andraschko et al. [23] frame smart home objectives to guide requirements and evaluation; De Franco et al. [24] outline research themes; Fakhr Hosseini et al. [25] propose terminology standardization; Mzili et al. [26] identify interoperability inhibitors (especially in healthcare); and Ezugwu et al. [27] categorize components and challenges in modern smart homes. However, each of these taxonomies either addresses broader smart home functionality, conceptual categorization, or terminological clarification rather than the practical functioning of interoperability mechanisms deployed in HA frameworks. Our work complements and extends these contributions by offering a taxonomy explicitly centered on interoperability as it materializes in real HA systems. Anchored in established models such as LISI and LCIM and empirically validated on operational open-source platforms, our taxonomy provides a more operational understanding of how interoperability is achieved, where it fails, and how future models can better support secure, adaptive, and semantically rich interaction across heterogeneous devices.

In addition to these distinctions, our taxonomy fills several methodological gaps that remain unaddressed in existing IoT and smart home classifications. Specifically, most prior taxonomies do not model interoperability as a multi-layer operational process, nor do they provide criteria for analyzing how interoperability mechanisms behave in deployed HA frameworks. By adopting an iterative conceptual–empirical methodology and validating the taxonomy against real open-source HA platforms, our work introduces a structured evaluative model that links interoperability concepts with their practical implementation. This approach enables a level of operational, mechanism-oriented analysis that is not offered by existing taxonomic solutions, thereby establishing the methodological contribution of our framework.

Beyond the need for semantic alignment and consistent integration mechanisms identified in HA-specific studies, insights from other IoT domains further illustrate how interoperability challenges generalize across ecosystems. These cross-domain, findings reinforce the importance of addressing data heterogeneity, semantic consistency, and privacy-aware processing when designing a taxonomy tailored to HA interoperability.

Recent studies in other IoT application domains have also highlighted the growing impact of data complexity, semantic consistency, and privacy-aware processing on interoperability. For instance, Kujur et al. [28] analyze model dependence in complex data classification, showing how data heterogeneity and feature variability influence the reliability of intelligent systems–a concern that similarly arises in the integration of heterogeneous devices and services within home automation. Likewise, Khan et al. [29] and Bashir et al. [30] address context-aware and privacy-sensitive IoT applications in agriculture, emphasizing the role of interoperable sensing and data management layers for adaptive and secure decision-making.

Additional studies in IoT ecosystems also address challenges that are closely aligned with interoperability concerns in home automation. Context-aware sensing and adaptive data processing, as explored by Khan et al. in IoT based monitoring scenarios [31], highlights the importance of consistent data representation across heterogeneous devices. Complementary work on decentralized and privacy preserving recommendation systems [32] underscores the growing need for secure and trustworthy data exchange within distributed IoT environments. Although these contributions are related to areas outside of HA, they highlight the need for shared methodological requirements, such as semantic alignment, reliable information flow, and user-focused privacy management, across interconnected IoT systems.

Although these studies address different domains, they share methodological principles relevant to home automation, such as the necessity of interoperable data acquisition, semantic consistency, and adaptive coordination among diverse devices. These parallels reinforce the generalized nature of interoperability challenges and solutions across IoT ecosystems, further motivating the design of a taxonomy tailored specifically to interoperability in HA.

To this end, we begin by providing a clear definition of interoperability in the context of HA, which serves as the foundation for the proposed taxonomy.

3. Home automation interoperability

Home automation enables devices that go beyond traditional functions [33], allowing appliances and systems to operate with varying degrees of autonomy. Such devices either respond to fixed, predefined parameters set by the user, or, with more recent technology advances, operate completely autonomously, dynamically adapting to changing environmental conditions. HA also facilitates the integration of traditionally separate devices and functions into a single, interoperable network, creating cooperative and increasingly intelligent systems within the home (see Fig. 2).

The concept of interoperability in HA refers to the ability of sensors and devices to communicate with each other, enabling seamless functionality across different ecosystems. In this domain, interoperability enables, for instance, the management of an irrigation system based on data from a weather station; the control of indoor and outdoor lighting based on Passive InfraRed (PIR)

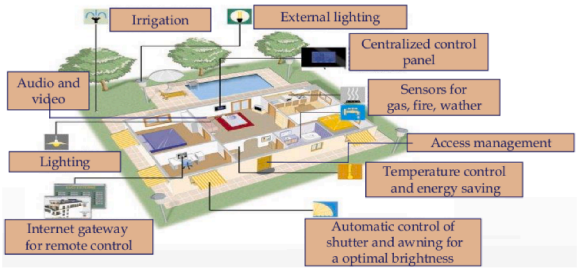

Fig. 2. Example of a home automation application.

Table 1
Most prominent standard technologies for home automation.

Standard Tech.	Media and Network	Main features	Main applications
DLNA	Multimedia protocols on IP by means of P2P or LAN networks	Designed only for media sharing among devices as Smart TV, Laptop, media player and smartphone	Audio/Video streaming (multimedia)
EnOcean	Wireless (Sub-GHz) by means of P2P and mesh networks (range 30-100m)	Enable devices to be powered by ambient energy sources such as light or motion using energy harvesting technology	Energy-efficient devices and sensors for smart homes
KNX	Wired (TP, PL, IP) and wireless by means of centralized Bus network (range up to 1km)	Widely adopted in both residential and commercial applications for controlling lights, HVAC, and others. It offers a high level of security and scalability. Requires professional installation and is not suitable for beginners.	HA and building control
LonWorks	Wired and Wireless (LonTalk) by means of a P2P network (range up to 1km)	Widely adopted in the industrial sector but less common in residential complex.	Industrial automation and building control
Thread	Wireless (Sub-GHz) and IPv6-based by means of mesh, local integrated network (range 10-30m)	Designed to overcome the limitations of Wi-Fi and Bluetooth. It supports low-power smart devices, and it is natively integrated in multiple ecosystems. Optimized for IoT with strong security (AES-128 encryption) and cloud integration capabilities	HA and IoT devices
UPnP	Multimedia network by means of a P2P over IP on local network	Mainly used for device discovery and sharing in LANs, Useful for networked devices that need to be automatically detected and shared	Device sharing over network
ZigBee	Wireless (2.4 GHz, SUb-GHz) by means of mesh network (range 10-100m)	Designed for battery-operated devices, making it energy-efficient. Wide compatibility with a growing ecosystem of devices.	HA and IoT

sensors and light level sensors; the automation of doors and windows based on the presence or absence of people in the home; the activation and deactivation of awnings and shutters based on information collected by both the weather station and light level sensors; the management of electricity flow by turning appliances on and off according to consumption levels, turning them off when consumption is high and on when energy costs are low (e.g., switching the washing machine and oven on and off according to specific time slots); and surveillance by means of a remote monitoring system consisting of cameras, microphones, and PIRs working in synergy, etc.

To date, several protocols have been developed to facilitate communication between devices. However, despite these efforts, there is still no widely adopted interoperability protocol for HA. The diversity of existing technologies suggests that a single standard may not emerge in the near future. Some representative examples, detailed in terms of features, media and networks, as well as main application field in Table 1, include: DLNA, EnOcean, KNX, LonWorks, Thread, UPnP, and ZigBee. All of the above technologies support different communication standards (e.g., Ethernet, Wi-Fi, Bluetooth) but remain incompatible with each other.

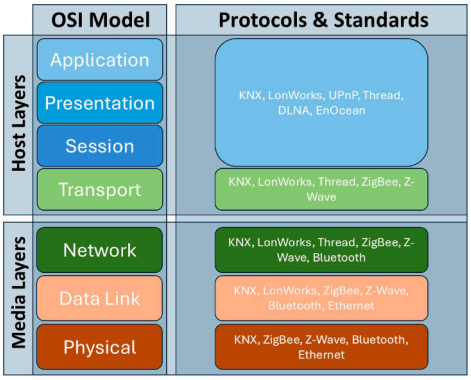

Fig. 3. Classification of home automation protocols & standards according to the OSI model.

The realization of the promise of HA technology is still hindered by industry efforts to impose proprietary systems. Some standards aim to create home networks, while others focus solely on managing them. Existing protocols and standards for HA networks can be ranked according to the OSI model, which classifies systems components according to the specific networking level at which they operate (i.e., physical, data link, network, transport and application).

For example, the KNX standard, which covers both the host and media layers, is present at every level of the OSI model, while low-level standards such as Bluetooth are only found at the network, data link and physical layers. An example of such a breakdown is shown in Fig. 3.

To achieve meaningful results in HA interoperability, a high level of abstraction between system layers is essential. This approach allows for a fully effective automation system that solves interoperability problems across standards and avoids reliance on ad hoc mappings between individual technologies. Such an infrastructure should implement a conceptual model to:

- Enable interoperability across diverse HA ecosystems.
- Support a range of smart ambient automation scenarios, from single home to smart city.
- Integrate heterogeneous devices in a unified data space, abstracted from underlying technologies.
- Remain generalizable, open, and compatible with existing smart home standards.

The ultimate goal is to implement solutions that allow users to introduce any HA device without worrying that it is incompatible with the current system.

A viable solution that meets the above requirements is software that ensures interoperability between different devices and services from different ecosystems. The most common approach in recent years has been to translate commands and services to enable different systems to communicate with each other. Such a solution typically employs software gateways, each responsible for translating information between the HA system (such as events, notifications and commands) and a common language that can be understood across platforms.

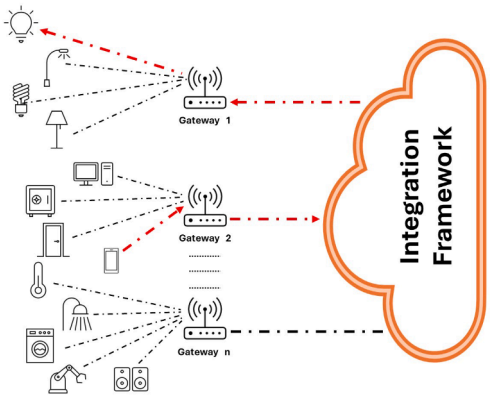

Fig. 4. Example of integration architecture.

When a device changes its state (e.g., a switch is pressed), the HA system generates an event that is captured by the gateway responsible for such a network. The gateway translates the event into a standard, unified message and forwards it to the integration framework. After receiving the translated message, the integration framework begins to process it through a system of rules and

generates a response to trigger the appropriate action on another device (e.g., switching on a lamp). The framework creates the new high-level message to invoke the function of the device involved and routes it to the appropriate gateway, which translates the response into another message in the other ecosystem language for execution. Fig. 4 shows an example of the aforementioned information exchange between devices belonging to different ecosystems.

The unique feature of such an approach is that responses are processed in real time and no database is required to store data to implement the mechanism. This real-time capability, combined with a common language, enables high flexibility in sharing data, functions and behaviors between devices. In addition, the approach does not rely on external systems, such as cloud resources, outside the local environment in which the devices are installed. To better understand this and other aspects, in the next section we detail a taxonomy of interoperability solutions in HA.

4. A taxonomy of interoperability solutions in home automation

4.1. Methodology

The taxonomy development was inspired by a systematic, iterative approach based on the methodology for taxonomy development proposed by Nickerson et al. [34]. This approach combined conceptual-to-empirical and empirical-to-conceptual iterations to ensure theoretical robustness and empirical relevance. The main interoperability dimensions were initially derived from an extensive review of existing classification models and interoperability frameworks, including LISI and LCIM, as well as their adaptations in the IoT and smart home domains. These models provided the conceptual basis for identifying recurring attributes and structural patterns of interoperability relevant to HA.

Subsequently, the taxonomy was refined through the empirical analysis of representative open-source HA frameworks. These frameworks were then used to validate the relevance, completeness, and distinctiveness of the identified categories. Each dimension and subcategory was explicitly defined to minimize overlap and ensure mutual exclusivity, guaranteeing that all identified interoperability features could be classified within the proposed structure. This iterative process enabled the taxonomy to remain descriptive and clear while accommodating a variety of architectural approaches and interoperability mechanisms.

The resulting taxonomy is designed to be dynamic and extensible. This enables the future integration of new technological paradigms, such as AI-driven automation and edge computing, as well as emerging interoperability standards (e.g. Matter and Thread). This adaptability ensures that the taxonomy remains aligned with the rapid evolution of smart home ecosystems while maintaining its methodological rigor.

It is worth noting that, although our methodology is inspired by and closely aligned with the iterative conceptual–empirical process proposed by Nickerson et al. [34], we adopted a streamlined version suitable for the scope and aims of this study. This allowed us to maintain methodological rigor without implementing the full formal procedure typically required for large-scale taxonomy development.

In its current state, the taxonomy (see Fig. 5) highlights the importance of protocol diversity, the role of standards and frameworks in enhancing compatibility, the need for robust integration platforms, the prioritization of security and privacy, and the varying levels of market adoption that define the HA landscape. Specifically, the taxonomy considers the following key dimensions:

- **Communication Protocols** - Communication protocols enable devices to interact with each other and can be divided into different categories based on their characteristics. At the first level, we propose a subdivision based on the medium, the communication layer and the type of data exchange. Such entries clearly differentiate how devices physically communicate and how data is structured and transmitted, providing a comprehensive view of the factors affecting protocol interoperability. The communication medium includes the physical and wireless technology through which data is transmitted, while the communication layer focuses on the software aspects related to data exchange and management. For the latter, we propose a tripartite subdivision into application protocols, transmission protocols, and transmission-hybrid protocols. This approach considers recent specifications (e.g., Matter) that introduce intermediate communication protocols, which are, in some cases, exclusively used for device-to-device handshaking. The Data Exchange entry, on the other hand, distinguishes between data that is shared directly and data that is shared in a disjoint manner, highlighting the nature of the collaboration between devices communicating within the HA network.
- **Standards and Frameworks** - In both HA and the IoT, standards and frameworks serve as the underlying architectures that enable devices from different manufacturers to work in synergy. While standards and frameworks share a common goal, they differ significantly in their functioning. A standard defines the rules and protocols that ensure devices can work together seamlessly, while a framework offers the development environment and tools for building IoT systems. The choice between open source and proprietary standards and frameworks has a significant impact on the flexibility, scalability and long-term sustainability of these platforms, determining how well they can adapt to evolving technologies and user needs. In the IoT, specifically, frameworks are critical to enable communication between different devices, facilitating broader system integration and interaction across multiple platforms. In the context of HA, instead, a further clarification should be made about the terms framework and middleware. While an HA framework is a software that collects basic tools and functionalities to manage smart devices and their data, an HA middleware sits at a lower level of a framework and acts as an intermediary between technologically different smart devices, applications or services, enabling them to communicate and share data. Middleware, while often considered frameworks in its own right, have distinct characteristics that merit separate consideration. In particular, the literature lacks an official, universally accepted definition of Middleware. To address this gap, we have introduced the Middleware Functionality category at the third level of the taxonomy, under the Frameworks entry. Ultimately, both standards and frameworks for ensuring interoperability

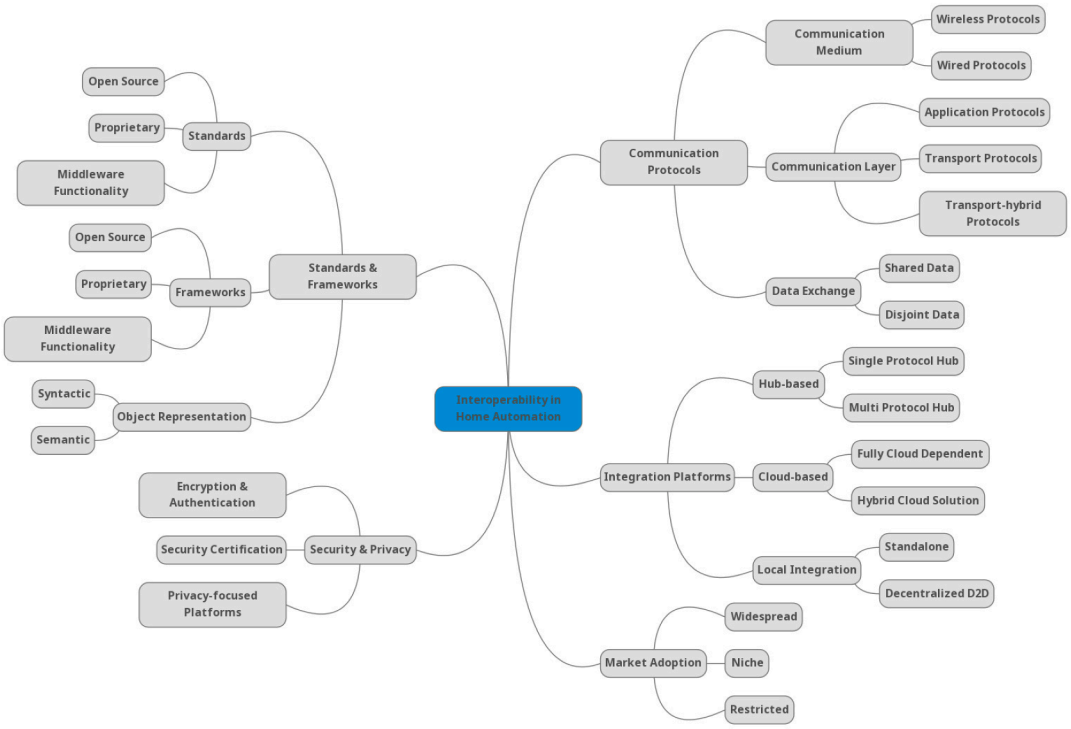

Fig. 5. Conceptual mind map of the proposed taxonomy for interoperability in home automation, illustrating its main dimensions and hierarchical organization.

define and make use of structures and formats for representing objects, which are entities that model devices and the services they provide. To this end, we have added the Object Representation entry, which includes the syntax and semantics used to describe devices and their attributes, as well as the relationships that are established between them.

- **Integration Platforms** - Integration platforms play a crucial role in orchestrating the communication and management of HA ecosystems. They can be thought of as the backbone that enables devices, protocols, and services to work together. Integration platforms provide the infrastructure for managing device communication, processing automation logic, and ensuring interoperability within a smart home ecosystem. Depending on their architecture and functionality, these platforms determine how devices exchange information, how data is stored and processed, and whether the system operates entirely within the home environment (i.e., with or without the need of hubs) or relies on external services. To this end, we have considered three primary categories for such an entry: hub-based platforms, cloud-based platforms, and local integration platforms. In addition, for each of the above subcategories, we consider whether the platform builds single- or multi-protocol hubs, whether it is a fully cloud-dependent platform or based on a hybrid cloud solution, and whether the local integration network configuration is standalone or based on a decentralized device-to-device logic.
- **Security and privacy** - Security and privacy are fundamental for their implications of connecting diverse smart devices safely. HA interoperability frameworks must address security and privacy across multiple layers. In this context, these concepts include platforms that implement robust encryption and authentication protocols, comply with security standard, and incorporate privacy-by-design principles. We distinguish three terminal categories: encryption and authentication, security certification, and privacy-centric platforms. These categories do not have subcategories and are therefore treated as final nodes.
- **Market Adoption** - Levels of market adoption serve as an evaluation of the influence of open and proprietary standards on HA market growth. The frameworks fall into one of three categories: widespread adoption, niche diffusion, or limited use. This classification stems from how connectivity frameworks and object representation affect ecosystem evolution and adoption trends within the HA domain. The market adoption subcategories do not include further subdivisions and are therefore designated as terminal nodes.

Fig. 5 shows the taxonomic model of interoperability in HA represented as a mind map, outlining the principal dimensions, categories, and their interconnections. The structure provides a comprehensive overview of the factors influencing interoperability across architectural, communication, and security layers.

In the following, we detail each of the above entries, providing a brief description and including examples that characterize each given set.

4.2. Communication protocols

4.2.1. Communication medium

The Communication Medium category refers to the physical or wireless technology used to transmit data between devices in the HA system. This dimension is essential because the medium used has a significant impact on the range, power consumption, bandwidth and infrastructure requirements of the smart home system. It distinguishes between:

- **Wired protocols**: Protocols based on physical wiring, such as Ethernet, KNX or Powerline Communication (PLC). These are often more reliable but less flexible and require extensive infrastructure set-up.
- **Wireless protocols**: Protocols such as Wi-Fi, Zigbee, Z-Wave, Thread, Bluetooth LE and LoRa that use radio frequencies to enable communication. These vary in range, power consumption and network topologies (e.g., mesh networking in Zigbee and Z-Wave).

4.2.2. Communication layer

The Communication Layer category focuses on the software aspects of how data is exchanged and managed once the physical medium (wired or wireless) is established. This dimension looks at the structure and rules that manage data transmission and provides insight into how protocols achieve interoperability at a technical level. We distinguish between:

- **Application protocols**: Such as MQTT, HTTP/HTTPS, CoAP, and WebSocket. These define how data is formatted, exchanged, and controlled at the software level. They facilitate interaction between devices and platforms and provide key functionality such as messaging (MQTT), RESTful web services (HTTP/HTTPS), and lightweight data handling (CoAP).
- **Transport protocols**: Such as TCP/IP and UDP. These govern the reliability and speed of communication, ensuring that data packets are transmitted correctly and efficiently. For example, TCP/IP guarantees reliability, making it suitable for critical data exchanges, while UDP prioritizes speed and is suitable for low-latency scenarios such as device status updates.
- **Transport-hybrid protocols**: Such as the Bluetooth Transport Protocol and the Matter Reliable Protocol. These protocols, represent a trade-off between traditional TCP and UDP and can be used in scenarios where a better balance between time-triggered and event-triggered communication is required. Specifically, the MRP introduces a security-enhanced version of UDP and offers a TCP-like, connection-oriented data transfer layer over a generic attribute profile.

4.2.3. Data exchange (device-to-device collaboration)

Data exchange is the process by which a device retrieves information coming from other devices. This is a critical aspect of the operation of an intelligent system because device responses are often conditioned by the status or measurements of other devices. There are two scenarios for data propagation:

- **Shared Data**: A shared data scenario is implemented by a protocol that natively allows information produced by a single device to propagate to all other devices on its network. This means that data related to a parameter measured by a sensor, such as the current temperature of an environment, or a change in the state of an actuator, such as the activation of a lamp, is propagated as is to all devices reachable via wired or wireless communication media. In this way, each device receives the information and reacts according to a programmed scheme, depending on its functionality.
- **Disjoint Data**: In a disjoint data scenario, the protocol does not natively allow data from a single device to propagate to others. Each device is functionally isolated, and data are typically collected by a third-party entity, such as a software framework. The framework processes and transforms the information for storage if needed and, eventually, determines the behavior of the other devices according to a programmed schema.

4.3. Standards and frameworks

4.3.1. Standards

A standard focuses on interoperability between devices from different manufacturers, ensuring that they are able to communicate seamlessly within the smart home ecosystem. Specifically, a standard in HA is a formalized set of rules, protocols, or specifications that define how devices and systems should interact to ensure compatibility, reliability, and security. In this context, we distinguish between open source and proprietary standards.

- **Open Source**: Open source standards are publicly available protocols designed to encourage widespread adoption by ensuring compatibility across devices and platforms. Examples are the Open Connectivity Foundation and ZigBee, which enable seamless interoperability between devices of different manufacturers. Open Standards promote transparency, reduce vendor lock-in and ease innovation, which make them the optimal choice for developing scalable and interoperable smart home ecosystems. Among the various open standards, it is worth mentioning Bluetooth and MQTT.
- **Proprietary**: Proprietary standards are managed by specific companies or organizations and typically limit interoperability to devices within their own ecosystem. Examples are the Apple HomeKit protocol and the Lutron's Clear Connect wireless technology. While proprietary standards often offer optimized performance within their ecosystems, they can limit flexibility, increase costs, and reduce compatibility with devices from other manufacturers. Notably, the Z-Wave protocol, which was originally proprietary, has lately moved to an open source model. Recently, some standards have adopted the royalty-free formula, i.e. a type of license that allows the use of the implementation of the specification with limited restrictions on its use and for an extremely low initial fee. An example of a royalty-free standard is Matter, from the Connectivity Standards Alliance.

4.3.2. Frameworks

A framework is software that provides a structured approach by collecting a set of tools and functionality to manage smart devices. Frameworks such as Home Assistant or OpenHAB provide platforms for integrating different devices and protocols, even if those devices follow different standards. As with standards, we distinguish between open source and proprietary frameworks. Since frameworks do not necessarily focus on enforcing universal compatibility between HA ecosystems, and in this respect they should implement an additional lower-layer software with middleware functionality, we describe Middleware functionality as a third-level category within this entry.

- **Open Source**: Open source frameworks provide extensible platforms for integrating a wide range of devices and protocols, facilitating the development of custom applications and automation. Some frameworks have been adopted by commercial entities, while others have been proposed by researchers as alternatives to proprietary systems or developed as prototypes for research purposes. Examples of commercially adopted frameworks include OpenHAB and Home Assistant, while DomoNet [35], IoTivity[2], and Dog Gateway[3] are research-oriented alternatives. Both categories of open frameworks allow users and developers to extend functionality and integrate different devices through community-developed modules or plug-ins. These frameworks generally support collaborative ecosystems that promote the inclusion of new protocols and devices, simplifying the development process by providing reusable code and services such as communication protocols, data handling, device management, and security features.
- **Proprietary**: Proprietary frameworks are usually controlled by a single company. They often offer a more sophisticated and user-friendly experience, but limit integration with third-party devices. Examples include Samsung SmartThings, Google Home and Amazon Alexa. While these frameworks prioritize ease of use, they tend to limit customization and scalability, forcing users to stay within the ecosystem of the manufacturer or face reduced compatibility with external devices. Recently, many major industry players, have shown increasing interest in Matter, a HA protocol that aims to address interoperability issues across different ecosystems. However, most proprietary systems still restrict Matter-compliant devices to a limited set of functionalities when managed by third-party applications, maintaining tighter control over their ecosystems and validating the above trend.
- **Middleware Functionality**: A HA Middleware works at a lower level within a framework, acting as an intermediary that enables communication and data exchange between technologically different smart devices, applications and services. Put simply, Middleware functionality facilitates the integration of different systems by managing the flow of data between them. A framework that implements Middleware functionality is defined as an *integration framework*. This perspective helps clarify the distinction between interoperability and integration frameworks, providing a more precise definition of Middleware functionality. An interoperability framework coordinates the systems it oversees to achieve predefined goals. In this context, each system operates autonomously and independently, allowing its unique features to be fully accessible. In contrast, an integration framework establishes a unified, overarching system that combines all the attributes of the systems it manages, while abstracting their technological differences. This unified system exploits the full capabilities of each constituent system, standardizing values, functions, and states into a common format to generate new functionality. Typically, these coordination capabilities require the use of third-party software functionality to establish the interrelationship between the existing systems in the form of a software layer, or Middleware [36], whose task is to mediate and coordinate all activities between such systems.

4.3.3. Object representation

The Object Representation category refers to the way in which the software interface models smart objects within the devices and services that constitute the system. Specifically, it describes the type of information collected and the methods used to characterize device functionalities and operational behavior. We distinguish between syntactic and semantic representation.

- **Syntactic**: The syntactic representation of objects defines the structures and formats in which devices and their attributes are described, focusing on the syntax rather than the intrinsic meaning of the data. For example, frameworks such as OpenHAB and Home Assistant use item-based models in which devices and their capabilities are syntactically represented by standardized key-value pairs. A consistent syntactic representation of entities through standardized data formats, such as JSON or XML, for instance, is critical for device interoperability. However, syntax alone does not provide the deeper semantic understanding required to handle interactions between devices.
- **Semantic**: The semantic representation of objects involves defining the meaning and relationships between devices, enabling intelligent and context-aware automation. Frameworks such as IoTSys and Dog Gateway support semantic models based on technologies such as Resource Description Framework, allowing devices to understand their respective roles and capabilities in a more meaningful way. Semantic modeling is key to complex automation scenarios where devices need to interact intelligently based on context rather than simple commands. Ontologies play a critical role in semantic representation, by defining a common vocabulary and set of relationships that describe the meaning and context of device attributes.

[2] http://iotivity.org/
[3] https://dog-gateway.github.io/

4.4. Integration platforms

Integration platforms are essential for orchestrating the interaction between devices and services in a HA system. They determine how devices communicate, how data is processed and shared, and where automation logic resides. Based on the architecture of the system, we distinguish three primary categories: hub-based platforms, cloud-based platforms, and local integration platforms.

4.4.1. Hub-based platforms

Hub-based platforms rely on a central hub to manage and coordinate smart home devices. To ensure interoperability they often, but not necessarily, support multiple protocols to provide a seamless and centralized control experience.

- **Single-protocol Hub**: A single-protocol hub is designed to manage devices using a single communication protocol, providing a streamlined but less flexible setup. An example is the Lutron Caséta, which implements only Lutron's proprietary protocol.
- **Multi-protocol Hub**: A multi-protocol hub supports multiple communication protocols (e.g., ZigBee, Z-Wave, Wi-Fi, and Bluetooth) and enables a wider range of device compatibility. Examples of this hub type include the SmartThings Hub, Hubitat Elevation, and Aeotec Smart Home Hub. The latter, for instance, is certified for ZigBee, Z-Wave plus, Wi-Fi, Thread and Matter.

4.4.2. Cloud-based platforms

Cloud-based platforms leverage cloud services to perform core functions such as device control, automation processing, data storage, and user interface (UI) rendering. These platforms offer easy setup and scalability, allowing users to remotely control and monitor their homes. However, their reliance on continuous Internet connectivity can introduce latency, reduce operational reliability during network outages, and raise privacy and data security concerns.

- **Fully Cloud-dependent**: Fully cloud-dependent platforms delegate all processing and automation logic to the cloud. These systems require an always-on Internet access for virtually all operations, including device control, system updates, and automation execution. Logic and processing are delegated from local devices, often thin clients, to remote servers. Such a model eliminates the need for complex local hardware and supports rapid updates and feature rollouts, but the system is strictly Internet-dependent and can easily be rendered inoperable or severely degraded during Internet disruptions. Examples of fully cloud-dependent platforms include Amazon Alexa, Google Home, and Apple Homekit (with iCloud).
- **Hybrid Cloud Solutions**: Hybrid cloud platforms integrate cloud-based services with local processing capabilities to balance scalability and resiliency. In such systems, the cloud is typically used for non-critical functions such as remote access, data analytics, and third-party service integration, while essential automation routines and control logic are executed locally. This architecture ensures that basic functionality - such as lighting control or security operations - remains available during Internet outages. Hybrid designs leverage the strengths of both cloud and local approaches to deliver advanced functionality without sacrificing operational continuity. Examples include Philips Hue (via the Hue Bridge) and SmartThings in hybrid mode.

4.4.3. Local integration platforms

Local integration platforms execute all control logic, automation routines, and data processing within the local network or directly on the smart home devices themselves. By eliminating reliance on the Internet or external cloud services, they enhance privacy, security, and operational resiliency. These platforms are well-suited for environments where low latency, data sovereignty, and autonomous functionality are essential, maintaining seamless operation even during network or Internet outages.

- **Standalone**: Standalone, locally integrated platforms centralize automation logic, device communication, and system management within a dedicated local controller or hub. These platforms typically run on dedicated hardware such as a Raspberry Pi, local server, or specialized appliance that hosts both the UI and the automation engine. While they may optionally support cloud connectivity for remote access or software updates, they are designed to operate independently of external services, giving users full control over their smart home ecosystem. This architecture provides a balance between local autonomy and centralized orchestration, with clearly defined limits to Internet dependency. Examples of platforms running on local servers are Home Assistant (with local installation), OpenHAB, and Domoticz.
- **Decentralized Device-to-Device**: Decentralized device-to-device, locally integrated platforms are systems in which devices communicate directly with each other using local protocols, without relying on a central controller or hub. In these types of platforms, automation logic is distributed among devices, often embedded in firmware or abstracted through peer-to-peer protocols (e.g., ZigBee, Thread, or Matter). Decentralization increases fault tolerance and limits single points of failure. However, it can also limit advanced functionality by requiring more complex device configuration. Examples of devices that enable this type of platform include thread-enabled Matter devices, Bluetooth mesh lighting systems, and Zigbee Green Power devices (such as switches and sensors).

4.5. Security and privacy

Home automation platforms need to address security and privacy at multiple levels to ensure reliable and trustworthy operation. In the context of HA, security and privacy refer to platforms that implement robust data encryption and authentication mechanisms, adhere to recognized security standards, and follow privacy-by-design principles. The current entry distinguishes between encryption and authentication, security certification, and privacy-focused platforms. None of these distinct categories have further sub-branches, so they are considered and discussed as terminal nodes of the taxonomy.

- **Encryption and Authentication**: This category includes platforms that implement robust data encryption (at various levels) and authentication mechanisms, such as secure network onboarding, device identity verification, and user access control, to protect communications and prevent against unauthorized access and manipulation by malicious actors. A HA platform that prioritizes security and privacy typically employs widely recognized encryption and authentication standards such as TLS/SSL, AES-128/256, and Public Key Infrastructure (PKI). For example, Apple HomeKit uses a secure remote password protocol and device attestation for end-to-end encryption between devices and controllers, the Nest ecosystem (Google Home) uses TLS for all cloud communications and OAuth 2.0 and two-factor authentication for account-level access, and Matter implements a mandatory encryption system for all communications.
- **Security Certifications**: This category includes platforms that comply with recognized security standards or certification frameworks that validate key aspects such as device behavior, firmware update policies, encryption protocols, and lifecycle management. These certifications help ensure that platforms meet baseline security requirements and follow best practices to protect users and infrastructure. Notable examples include certifications from the IoT Security Foundation, compliance with ETSI EN 303 645, and frameworks from industry alliances such as the Wi-Fi Alliance (e.g., WPA2/WPA3), Matter, and UL IoT Security Rating.
- **Privacy-focused Platforms**: Privacy-focused platforms are those HA ecosystems that are built with privacy-by-design principles. This means minimizing data collection, processing data locally without relying on remote processing services, avoiding unnecessary cloud storage, and giving users transparency and control over the data they consume. Examples of privacy-focused platforms include Home Assistant, which by default processes all automation and data locally without requiring Internet access, and Hubitab Elevation Hub, whose philosophy is local-first with no mandatory cloud connection and user data remaining on the device.

4.6. Market adoption

This entry examines the impact of open and proprietary standards on market growth, distinguishing between widespread adoption, niche diffusion, and restricted use of HA platforms. The rationale for this subdivision lies in how factors such as connectivity and object representation influence adoption rates and ecosystem development within the HA sector. As these categories are not further subdivided, they are considered terminal nodes within the taxonomy structure.

- **Widespread**: This category includes platforms and standards that have achieved significant market penetration and are commonly supported across a broad spectrum of devices, vendors, and use cases. These solutions are often based on open or widely adopted specifications and promote interoperability. The widespread adoption of these technologies is largely due to their use of standardized, well-supported connection types (e.g., IP-based networking, IEEE 802.15.4) and their implementation of flexible, abstract device representation models that simplify integration and enable rich interactions between heterogeneous devices. Technologies such as Wi-Fi, BLE, and ZigBee fall into this group due to their long-standing use in both consumer and industrial settings. Platforms such as Google Home and Amazon Alexa have also achieved widespread adoption by offering developer-friendly ecosystems and broad device compatibility, enabling rapid user adoption. Matter is a more recent but impactful addition, emerging as a collaborative standard supported by major tech companies to unify fragmented HA protocols and ensure device interoperability. Although Samsung SmartThings and KNX originally had more focused or professional use cases, they have expanded their reach to a broader audience, warranting their inclusion in this category as well.
- **Niche**: Niche diffusion refers to platforms and standards that are well developed and used in specific markets or among enthusiast communities, but have not achieved widespread adoption. Technologies in this category often rely on more specialized connectivity models or impose rigid abstraction layers for device representation that, while powerful, require deeper integration efforts that slow broader ecosystem adoption. These technologies are often limited by ecosystem boundaries, technical complexity or access restrictions. Apple HomeKit is an example of this category: while it offers high-quality integration and privacy features, it remains constrained by strict certification requirements and limited developer access, limiting its broader market adoption. Similarly, Thread and LoRa address specific needs (Thread for low-power, low-latency mesh networking and LoRa for long-range communications in rural or large-scale installations), making them valuable but specialized. OpenHAB, an open source platform, is a notable example of an initiative that has moved from restricted use to niche diffusion. It has cultivated an active community and expanded its device support, demonstrating how open projects can mature into stable, mid-level solutions without full commercial support.
- **Restricted**: Restricted use refers to platforms and technologies that remain largely confined to experimental, academic, or narrowly defined environments. These platforms often experiment with non-standard or evolving communication protocols and introduce highly flexible but complex device abstraction models that, while powerful for research, hinder ease of deployment and limit interoperability in consumer environments. These solutions are typically developed for research, prototyping or testing purposes, often as open source projects, but lack the sustained development, documentation, or ecosystem support necessary for broader adoption. Examples include Dog Gateway, Domonet, Freedomotic, IoTivity, IoTSys, and Thing System. While these platforms may offer valuable innovations or flexible architectures, they are generally not intended for or suitable for mainstream consumer use. Their use is often limited to university labs, small-scale pilots, or specialized industrial scenarios, and they may eventually be deprecaded or absorbed by more robust systems.

Although the current taxonomy primarily focuses on interoperability mechanisms and architectural integration layers, it has been designed as an extensible framework that can evolve alongside emerging technologies, such as AI-driven automation, edge intelligence, and hybrid edge-cloud architectures. Future refinements, may incorporate new categories or sub-dimensions to reflect these advances and their implication for interoperability in next-generation smart home ecosystems.

In the next section, we validate the proposed taxonomy by applying it systematically to a selection of representative open-source home automation frameworks. Open-source solutions were chosen intentionally, as they allow transparent access to architectural components, communication protocols, and data exchange mechanisms, facilitating a more comprehensive and reproducible analysis of interoperability dimensions. While proprietary systems dominate the commercial smart home market, the restricted documentation and closed architectures of these systems often prevent meaningful comparison at the structural level.

However, it is important to note that the taxonomy itself was designed to be license-neutral and can accommodate both open-source and proprietary frameworks equally well. Its classification principles and hierarchical organization are flexible enough to capture interoperability characteristics across diverse ecosystems. The current evaluation demonstrates the descriptive validity and analytical utility of the taxonomy, and future work will extend this validation to include proprietary and large-scale industrial solutions.

Finally, despite their relatively limited maturity, open-source systems often demonstrate greater adaptability and potential for community-driven innovation. This makes them ideal for initial validation of a taxonomy intended to support the evolution of open, transparent and interoperable home automation ecosystems.

5. Analysis of key open-source interoperability frameworks for home automation

This section illustrates the practical application of the proposed taxonomy by analyzing a representative set of open-source interoperability frameworks for HA. Initially, four prominent platforms (i.e., **OpenHAB, ioBroker, Domoticz and Home Assistant**) were selected based on the ranking provided by Setz et al. [37], which evaluates HA solutions according to a set of interoperability and architectural criteria. Each framework takes a different approach to device integration, automation logic and system design.

To enhance the robustness and representativeness of the comparative analysis, three additional open-source frameworks (i.e., **IoTivity, DomoNet,** and **Dog Gateway**) have been incorporated alongside the original four platforms. While the initial selection emphasized maturity and adoption, the newly added frameworks were chosen to broaden the spectrum of interoperability paradigms encompassed by the analysis. Specifically, IoTivity represents a standards-based approach that emphasizes communication and service layer interoperability, DomoNet exemplifies an academic ontology-based model centered on semantic integration, and Dog Gateway illustrates a middleware-oriented solution that bridges the gap between research prototypes and deployable systems.

We examine the capabilities of all seven frameworks in areas such as local versus cloud-based control, object representation, security mechanisms, and extensibility using the taxonomy as a structured analytical tool. Including these additional frameworks enables us to validate the proposed taxonomy more comprehensively, demonstrating its applicability and discriminatory power across systems that differ in architecture, maturity, and interoperability strategy. This expanded analysis highlights the strengths and limitations of current solutions, providing insights to guide the development of future interoperable, sustainable platforms.

5.1. OpenHAB

OpenHAB is an open source HA integration framework based on the OSGi Alliance specifications [38], initiated in 2010 by a group of developers. Leveraging the OSGi architecture, OpenHAB structures its functionality into modular bundles, allowing extensibility through add-ons that provide new UIs, automation logic, or support for additional devices. To manage the runtime environment and control the deployment and lifecycle of these bundles, OpenHAB uses Apache Karaf [39], a lightweight container that extends OSGi-based systems with dynamic module loading, shell access, and provisioning support.

5.1.1. Architecture

The OpenHAB architecture (see Fig. 6) consists of three main components: the Core, the Binding Provider, and the UI and Services layer. The Core is responsible for essential functions such as the message bus, data storage and logging. For example, it provides flexible state persistence for each element, which is particularly useful for recovering system states after a crash. The Binding Provider component handles communication with external devices and services, allowing OpenHAB to integrate with a wide range of hardware and protocols. The UI and Services layer includes several key features: a rules engine for defining device behavior and ensuring interoperability; a markup-based system for creating custom UIs; and a REST API that allows third-party applications to interact with the OpenHAB framework.

5.1.2. Communication protocols

Concerning *communication medium*, OpenHAB can connect to IP-based smart switches, routers, and hubs via Ethernet (LAN) when they provide network APIs such as REST or MQTT over TCP/IP. It also supports interfacing with serial devices (USB or RS232) through specific bindings. For wireless communication, OpenHAB can interact with IP devices on a wireless LAN (e.g., smart bulbs and thermostats) using Wi-Fi. Additionally, it employs USB dongles or gateways to support mesh wireless protocols operating at 2.4 GHz (Zigbee) or sub-GHz frequencies (Z-Wave). OpenHAB further supports Bluetooth/BLE-enabled devices via appropriate bindings, as well as other proprietary wireless protocols through gateways or bridges.

About *communication layer*, OpenHAB covers the application and transport layers. It supports several protocols, including MQTT, REST APIs, CoAP, the Zigbee Cluster Library, Z-Wave Command Classes, and custom protocols via specific device bindings, such as those for KNX, Modbus, and EnOcean. MQTT and HTTP/REST typically use TCP as the transport protocol for IP-based protocols, while CoAP uses UDP. Zigbee and Z-Wave manage transport through their own mesh network protocols that operate over wireless media. Serial communication with devices is handled via UART or USB transport. For external integration, OpenHAB exposes REST, Server-Sent Events (SSE), and WebSocket APIs.

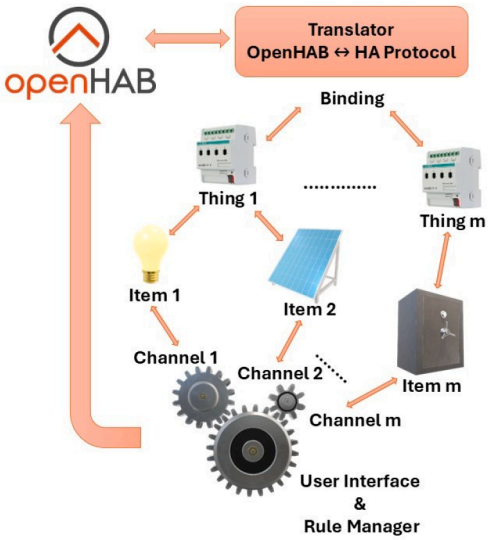

Fig. 6. OpenHAB architecture.

Regarding *data exchange*, OpenHAB maintains a centralized state model that keeps a shared *Item* repository reflecting device states. However, in some cases, it only sends commands without receiving device-state feedback (write-only control), and devices independently push updates. Additionally, some bindings use polling to update states asynchronously, which can result in a temporarily disjointed data.

5.1.3. Standards & frameworks

OpenHAB is a *open source framework* with *middleware* capabilities. Although it is not a formal standard, it supports and integrates numerous open and proprietary de facto HA standards. As middleware, it abstracts heterogeneous devices through a unified model of *Things*, *Bindings*, *Channels*, and *Items*, enabling protocol translation (e.g., Z-Wave to MQTT) and centralized orchestration via rules, scripts, or the UI. In terms of *object representation*, OpenHAB separates the physical aspect (device communication and addressing) from the functional aspect (operations exposed for automation and UI interaction). Bindings act as protocol adapters that translate device-specific formats into OpenHAB's internal model. Each Thing, representing a physical device or logical service endpoint, requires a suitable binding (e.g., Z-Wave binding) and defines Channels, which are the data/control interfaces. Items are logical abstractions linked to Channels and serve as the system's core functional units. A single Item can be associated with multiple Channels, even across different Things, enabling multi-source inputs or multi-device control. All these components (i.e., Things, Channels, Items, and Rules) can be described in terms of their syntax and semantics. Things are uniquely identified by UIDs and defined via YAML files or the UI. Semantically, they abstract devices or services, with Channels mapping to specific data points or functions (e.g., temperature, switch state). Items, defined in ".items" files, represent the unified data model of the system. They encapsulate state and control logic and are central to rule execution and UI interaction. Semantically, they unify heterogeneous technologies under a consistent abstraction. Rules encode automation logic and temporal behavior over Items, and are defined using a domain-specific language, JavaScript, or Blockly, a visual programming language that uses blocks to create complex scripts.

5.1.4. Integration platforms

Concerning the *integration platform* aspect, OpenHAB is best characterized as a *local integrated platform* that can optionally include cloud services. It is designed to run locally on a user-managed server (e.g., Raspberry Pi or Docker) and performs automation and device management without requiring an internet connection. Although it is *not a fully decentralized device-to-device* mesh system, OpenHAB can communicate directly with devices using local protocols (e.g., Zigbee via USB sticks, MQTT, KNX, etc.) and supports peer-to-peer or bridge-like setups. It also acts as a *multi-protocol hub* by supporting a wide range of device types and communication protocols through bindings. These bindings unify diverse protocols into a common model, enabling users to integrate and automate across various vendor ecosystems. Finally, while OpenHAB is not cloud-dependent, it offers optional integration with the myOpenHAB cloud service for remote access, push notifications, and connectivity with third-party services such as Amazon Alexa and Google Assistant.

5.1.5. Security & privacy

In terms of *security and privacy*, OpenHAB offers partial support for encryption and authentication. It enables HTTPS for the web interface and REST API through Jetty, although secure communication must be manually configured via SSL certificates. Since version 3, OpenHAB includes built-in user management with role-based access control, supporting basic username/password authentication

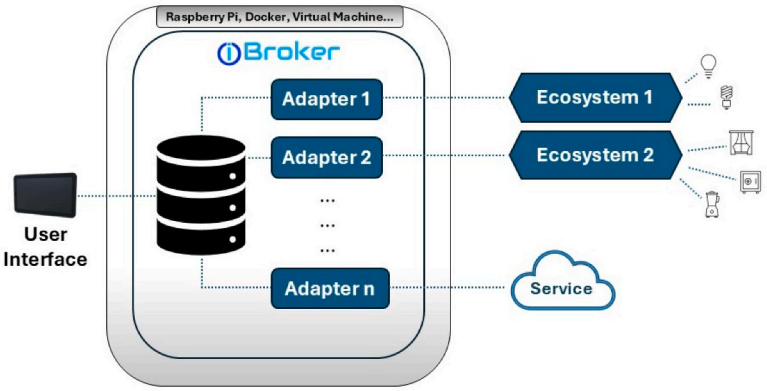

Fig. 7. Abstraction of the ioBroker architecture: UI, storage, adapters and system management.

and API tokens for service-to-service communication. However, integration with external identity providers (e.g., OAuth2) is not natively supported.

OpenHAB, as an open-source project, relies on community-driven best practices and has not undergone formal security certification. The responsibility for securing the host environment and managing system-level protections is delegated to the user. A notable strength lies in its privacy-preserving architecture: OpenHAB operates entirely on local infrastructure, with no inherent cloud dependency. Data is not collected or transmitted to external servers unless explicitly configured by the user. Full control over configuration, logs, and system state remains with the user. Nonetheless, sensitive information requires manual protection, as encryption mechanisms are not enabled by default.

5.1.6. Market adoption

Regarding *market adoption*, OpenHAB exhibits a niche to moderate presence, primarily attracting technically proficient users seeking local control, protocol interoperability, and high level of configuration.

5.2. ioBroker

ioBroker is an interoperability IoT middleware based on the Fog computing paradigm integration both free and commercial products. It is characterized by its modularity and plugins support, named adapters. ioBroker has a scalable structure permitting to connect several ioBroker servers to create a multihost system form.

5.2.1. Architecture

Fig. 7 illustrates the ioBroker architecture, showing how the system can manage multiple ecosystems and services. Its architecture is based on modular *adapters*, which are JavaScript plugins for Node.js. These adapters enable communication with specific devices, device categories, manufacturers, or external services (e.g., querying web APIs). Through adapters, ioBroker connects physical devices such as smart sockets, relays, and temperature sensors to the central system. They facilitate both data acquisition and device control, often over WLAN within the home network. Additionally, ioBroker supports a multilingual UI, aided by a built-in language assistant.

5.2.2. Communication protocols

ioBroker supports both wired and wireless communication. For wired connections, it operates over Ethernet, enabling the interaction with devices via IP-based protocols. It also allows integration with serial interfaces, such as RS232 or USB, via specific adapters. On the wireless front, ioBroker is compatible with several technologies such as Wi-Fi, ZigBee, Z-Wave, and enOcean via USB dongles. It partially supports for Bluetooth/BLE, and the integration of proprietary wireless systems via bridges or gateway adapters.

From a *communication layer* perspective, among other application protocols, ioBroker supports MQTT, HTTP/REST, Modbus, CoAP, OPC-UA, and BACnet. The underlying transport protocols are primarily TCP/IP and UDP even if, depending on the adapter, hybrid transports such as WebSockets and CoAP over UDP may also be supported.

In terms of *data exchange*, ioBroker centralizes and hierarchically organizes data from all integrated devices, enabling in this way the communications among adapters. This permits cross-adapter interoperability and unified automation logic.

5.2.3. Standards & frameworks

In terms of *standards and frameworks*, ioBroker is a fully *open-source* platform distributed under the MIT license. Despite its open nature, the system supports integration with proprietary protocols (e.g., Homematic or KNX) via dedicated adapters. ioBroker operates functionally as a comprehensive *middleware* layer, that abstracts the underlying protocol complexity through adapters and exposes a unified internal object model to enable consistent interaction across devices. Automation and orchestration logic can be implemented using either the built-in Javascript engine or the visual Blockly environment, providing flexibility to users of all levels.

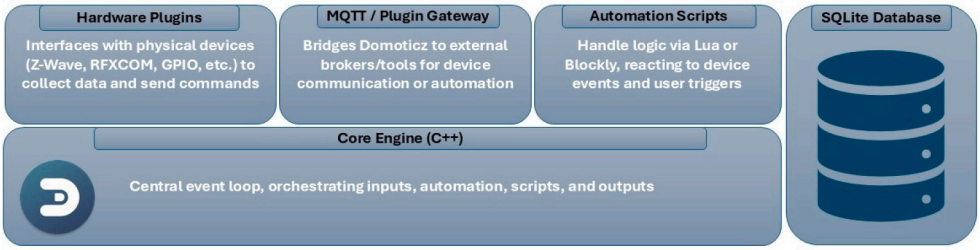

Fig. 8. Domoticz core architecture.

ioBroker uses a hierarchical ID namespace to organize and *represent objects* within syntactic structures. The platform semantically enriches each object with descriptive metadata fields, which, together with roles, define function and context, supporting semantic interpretation by dashboards and automation tools.

5.2.4. Integration platforms

Because of its stand-alone nature, ioBroker is primarily designed for *local integration* and can entirely operate offline but with limitations. It also supports hub-based integration through its adapter-based *multi-protocol* architecture. Cloud-based integration is optional and can be selectively enabled for specific services, without requiring full cloud dependencies.

5.2.5. Security & privacy

Security in ioBroker is only partially addressed. While the platform supports HTTPS and user authentication, these features require manual setup. By default ioBroker runs locally, making it privacy-preserving by design. However, encryption for external API access must also be manually configured, and the system lacks native support for external identity providers (e.g., OAuth2). As a result, its use is generally limited to technically skilled users.

5.2.6. Market adoption

ioBroker has gained significant popularity within the wider open-source and DIY smart home communities. Its extensive interoperability features, modular architecture and wide range of device adapters, developed by an active contributor ecosystem, are particularly appreciated by technically proficient users. While it remains largely adopted by enthusiasts and researchers rather than commercial entities, ioBroker is a mature and flexible platform that effectively showcases the potential of community-driven innovation in advancing interoperability within home automation systems.

5.3. Domoticz

Domoticz is open-source HA software writte in C++ programming language. It is designed to configure, monitor, control, and automate smart devices in the home. It implements a web server where users can interact with the system through a web interface or the mobile app available both for Android and IOS.

5.3.1. Architecture

Fig. 8 shows the core architecture of the framework. It comprehends the Core Engine where events, inputs, automations and outputs are managed. To do that, it takes avantage of Hardware plugins to interface physically devices, an MQTT plug-in gateway and an automation script module to implements logic. While Lua and Blockly are natively supported for handling automation logic, the implementation of other automation systems (such as Node-RED) must be interfaced externally. Its functioning come up beside a SQLite Database.

The Domoticz system can run locally on inexpensive hardware, such as a Raspberry Pi, within a user home network.

5.3.2. Communication protocols

It can interface with wired and wireless devices, connecting them directly or through gateways, including all appliances related to them. The framework supports smart devices such as lights, switches, smart sockets, thermostats, alarm systems, curtains, roller shutters, and IP cameras. It also supports internet services such as Google Pub/Sub, publishing to IoT sites, InfluxDB, MQTT, etc. The system can also send notifications and alerts to any mobile device when user-defined conditions are met. It implements both application and transport layers and disjoint data exchange.

From a consumer's perspective, Domoticz uses widgets to represent devices on the graphical UI. Domoticz displays devices according to their typology (e.g. thermostat, switch, dimmer, door, camera, bell, wind meter) and each device belonging to a typology may have one or more widget to represent it and activate its functionalities. Domoticz support the creation of Python plugins to extend support to new devices and hardware, furnishing C++ API usable throuth HTTP/HTTPS and WS/WSS protocols.

5.3.3. Standards & frameworks

Domoticz implements middleware functions and it natively supports several smart devices and smart home protocols. It offers native support for many technologies and ecosystems, such as X10, Philips SBC, EnOcean, MyHome, OpenWebNet, MQTT, Netatmo, Nest, Raspberry Pi GPIO interfaces, HTTP/HTTPS pollers, LoRa, Z-Wave, and Apple HomeKit. This broad compatibility reflects Domoticz's commitment to interoperability across legacy and modern platforms, enabling integration into various home automation environments with minimal customization.

5.3.4. Integration platforms

Domoticz works locally and do not take advantage of cloud services.

In terms of *interoperability aspect*, Domoticz supports the creation of automation and scripts to ensure interoperability among its supported devices. To this end, the system interfaces with sensors and actuators via the MQTT protocol, standardizing the functionalities and access methods for each device type. Domoticz offers the possibility to create trigger that activate the functions of one or more devices according to certain conditions, such as the expiration of a timer or a system notification. It also allows users to aggregate devices into groups to invoke a function valid for multiple devices and create scenes, or predefined configurations of different devices for specific situations. Additionally, Domoticz allows users to create Python, LUA , or dzVents scripts to program device behavior and manage system events. Similar to OpenHAB, Domoticz uses Blockly to create scripts via a graphical UI.

5.3.5. Security & privacy

In terms of *security and privacy*, Domoticz protects its graphical UI with username and password. It supports multi-user access to the system and allows each user to have different rights through roles. Specifically, it allows for admin users with access to configurations, settings, and user management; user that can control devices, such as turning on or off lights; and viewers that can access devices' views but cannot control them. However, Domoticz supports multiple identification and authorization methods. In addition to logging in via a browser, Domoticz supports logging in via an access token (JSON Web Token) or basic authentication credentials over HTTP for API calls. Access is permitted even without any credential if the request comes from a trusted network. Domoticz also supports OAuth2 and OpenID Connect, allowing user to protect access to one or more devices.

5.3.6. Market adoption

Domoticz primarily appeals to advanced users and developers who value fine-grained configuration and cross-platform flexibility, especially on lightweight embedded platforms like the Raspberry Pi. However, it has not gained widespread adoption among general consumers due to its relatively steep learning curve, less intuitive user interface, and lack of user-friendly abstractions. These factors hinder its mainstream penetration compared to home automation solutions that are more consumer-oriented and emphasize ease of setup and integrated ecosystem support.

5.4. Home assistant

Home Assistant is an open-source system written in Python and it is supported by a global community of contributors. Thanks to their efforts, the system integrates a large number of devices, protocols, and UIs. There are two main ways to install Home Assistant: (1) as a stand-alone operating system designed to run the ecosystem on a single-board computer (e.g., a Raspberry Pi) or a virtual machine, or (2) as a container on an existing operating system (e.g., via Docker). The primary difference between these approaches is that the stand-alone installation supports add-ons, which are not permitted in the containerized version. Home Assistant is also distributed as a pre-installed image on dedicated, pre-configured hardware.

5.4.1. Architecture

Fig. 9 illustrates the Home Assistant architecture, as described in the official documentation. The diagram highlights its main components, including: *State Machine*, which tracks changes in device states and notifies the Event Bus accordingly; *Event Bus*, through which events are transmitted and services are invoked; *Service Registry*, where components register their available services; and *Timer*, which generates time-based events for scheduling purposes.

Devices are represented using syntatical approach.

5.4.2. Communication protocols

Home Assistant supports wired and wireless communication protocols, including the *application*, *transport*, and *hybrid* layers. Data exchange follows a disjoint model where interactions are governed by distinct communication events rather than shared data spaces. Through its event-driven architecture, Home Assistant allows devices and automation routines to dynamically respond to specific triggers, such as sensor updates or user-defined conditions, by invoking predefined functions that execute corresponding actions or actuators. This design ensures flexibility and scalability when managing heterogeneous devices and services in both local and cloud environments.

5.4.3. Standards & frameworks

To extend the range of supported devices and technologies, Home Assistant can leverage the use of add-ons. Add-ons extend the default functionalities of the system and enable support for additional HA technologies and third-party services. Some add-ons are officially maintained, while many others are developed and shared by the community. Some example of supported devices and

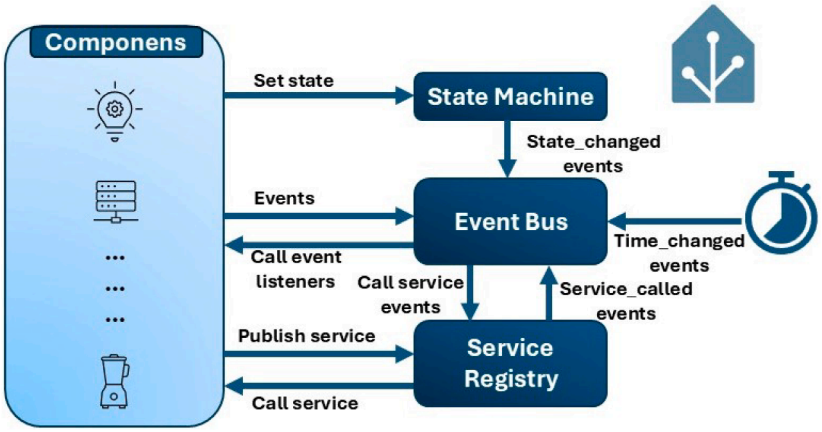

Fig. 9. The home assistant architecture.

technologies are: Amazon Alexa, ESPHome, Google Assistant, Google Cast, HomeKitBridge, Ikea Tradfri, KNX, Leviton Z-Wave, Lutron, Matter, MQTT, Philips Hue, Shelly, Samsung Smart Things, Sonons, Z-Ware and Zigbee.

A Home Assistant device is a physical entity that is equipped with one or more sensors or actuators. The dashboard provides a real-time overview of active devices and services. New devices or services can be integrated via the UI or by editing configuration files, typically using procedures specific to each add-on. For instance, the KNX add-on allows users to import ETS configuration files and map KNX group addresses to specific functionalities. However, creating and assembling devices remains manual. Users must map imported functions to device instances according to a predefined schema for each device category.

5.4.4. Integration platforms

Home Assistant works locally, but it can connect to other cloud-based services to implement and integrate services furnished by other entities. Home Assistant achieves interoperability through an automation model based on the classic "if-then" paradigm. This model enables specific operations to be executed when predefined conditions are met. These conditions may include device states, time-based triggers, or a combination of both. Users can also define scenarios, which are sets of actions triggered by a single command. For more complex behavior, Home Assistant supports scripts, which allow users to compose advanced automation routines. The ecosystem can be controlled via a smartphone app or web interface.

5.4.5. Security & privacy

In terms of *security and privacy*, Home Assistant processes data locally and only stores personal data when necessary for specific functionalities. Cloud-based features are disabled by default and are only activated with explicit user consent. Beyond this local-first approach, Home Assistant does not implement additional built-in security or privacy strategies. Instead, it relies on recommended best practices, such as using VPN services, creating strong passwords, and enabling two-factor authentication to secure system access. To minimize the risk of introducing insecure or bugged code, it is also advisable to use official add-ons rather than community-developed ones. System updates are made available through the UI when released.

5.4.6. Market adoption

In terms of *market adoption*, Home Assistant is widely recognized and actively used by a broad community of hobbyists, practitioners, and researchers. Its popularity stems from its flexibility in managing diverse sensors and actuators, as well as the project's clear focus on usability and accessibility, which makes it appealing beyond technically skilled users.

5.5. IoTivity

IoTvity[4] is an open-source domotic interoperability framework aiming to be a new business standard by creating a robust and extensible domotic architecture for smart devices. It follows the Open Interconnect Consortium (OIC) specifications of the Open Connectivity Foundation (OCF) [40]. The OCF specifications are agnostic regarding the application domain and define a generic model and common standardization according to the IoT paradigm to represent resources and their access through standard interfaces for discovery, presenting of services, communication and security.

[4] Open Connectivity Foundation. [n.d.]. IoTivity website: https://iotivity.org/

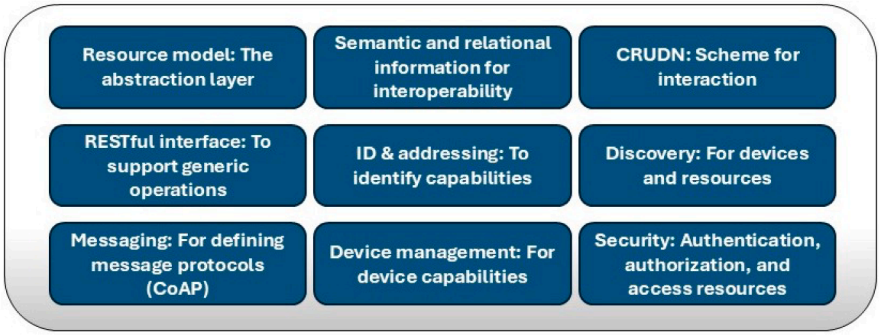

Fig. 10. IoTivity core components.

5.5.1. Architecture

The IoTivity framework includes the following core components (see Fig. 10):

- *Resource model*: It provides an abstraction layer to model resources and the way they interact.
- *Semantic and relational information*: They specify the semantic and relational information required to implement interoperability in a contextualized environment and provide for protocol independence of communications in the model.
- *CRUDN*: It provides a generic scheme for the interactions between devices using the client-server paradigm. The client is the device that launches a RESTful operation, while the server is the recipient device. Any device that exposes entities as resources can be a server.
- *The RESTful interface*: It supports generic operations, such as Create, Retrieve, Update, Delete and Notify.
- *ID & addressing*: They define the identification and addressing capabilities using URI syntax, such as $'ocf :< authority > // < path >? < query >'$.
- *Discovery*: It defines the process for discovering devices and their resources through URI and RESTful operations.
- *Messaging*: It provides specific message protocols for RESTful operations. IoTivity adopts the Constrained Application Protocol (CoAP) defined by the Internet Engineering Task Force (IETF) as the main IoT connectivity protocol. CoAP is a light-weight hypertext transfer protocol that can easily translate into HTTP. IoTivity also provides mechanisms to include other transport protocols.
- *Device management*: It specifies the rules for managing the capabilities of devices, as well as their initial setup, monitoring, and diagnostics.
- *Security*: It includes authentication, authorization, and access control mechanisms required for secure access to entities.

5.5.2. Communication protocols

IoTivity supports wired and wireless communication media and provides a comprehensive stack spanning the application and transport layers of the IoT protocol hierarchy. Built upon standard internet protocols, it primarily leverages CoAP over UDP and, when necessary, TCP for reliable delivery. The framework defines a resource-oriented model in which devices expose and consume resources via RESTful interfaces, enabling uniform communication across heterogeneous systems.

Data exchange within IoTivity follows a structured, rule-based model coordinated by a centralized resource directory that manages device discovery, registration, and invocation. This design ensures interoperability through standardized message formats and consistent resource access semantics, rather than ad hoc communication. Additionally, IoTivity's modular architecture supports integration with other IoT standards, including OCF specifications, enabling seamless communication across compatible devices and networks.

5.5.3. Standards & frameworks

IoTivity is an open-source framework that provides middleware functionalities encompassing network, syntactic, and semantic interoperability, along with device representation. In alignment with the OCF specifications, IoTivity defines a structured model based on the concepts of entities, resources, URIs, resource types, properties, representations, interfaces, collections, and links.

A resource represents an *entity* (i.e., a physical device) and is accessible via a URI, supporting interactions and operations following the REST architectural style. The properties of a resource are defined as key–value pairs, and each resource instance derives from a specific resource type. A one-way relationship between two resources is defined as a link, while a collection is a resource that aggregates references (properties and links) to other resources. Collections may represent hierarchies, groups, or indices, allowing related resources to be managed collectively under a single URI.

The state of a resource is determined by a defined set of properties. IoTivity employs a dynamic discovery process to import and manage resources, allowing devices to connect to or disconnect from the framework autonomously through a registration mechanism.

An *endpoint* represents the source or destination of requests exchanged between resources. Each device can be associated with one or more endpoints to support request–response interactions. The OCF specifications define three distinct discovery mechanisms for identifying available resources within the network:

- *Direct discovery*: The device that is hosting the resources publishes them locally to enable their discovery through peer inquiry in a unicast or multicast CoAP message. The server that receives the request sends a response with the discovered information directly back to the requesting device.
- *Indirect discovery*: The device hosting the resource is neither the requesting client, nor the device that is providing or publishing the information for discovery; instead, the information is published by a third device that acts as intermediary. The client making the discovery may send a unicast or multicast discovery request to the device hosting the discovery information.
- *Advertisement of discovery*: The device that is hosting the resource is the same as that which enables discovery.

5.5.4. Integration platforms

IoTivity operates primarily on local networks but can also leverage cloud-based services and functions. It enables both device-to-device and device-to-network communication by supporting multiple protocol stacks, including BLE, NFC, Bluetooth, and IPv4/v6, thus providing compatibility with both wired and wireless connections.

IoTivity employs the OCF Bridging specifications to interconnect devices and resources, defining mechanisms for linking OCF and non-OCF implementations. In this context, OCF Bridge devices act as network entities that use bridged protocols instead of native OCF communication. These bridges serve as gateways that expose non-OCF clients and servers to the OCF ecosystem by creating virtual OCF resources.

To manage device behavior, the OCF standard introduces the concepts of *Scenes* and *Rules*. A Scene automates operations by grouping user-defined configurations into a single client operation, representing a static entity that stores predefined property values across a collection of resources possibly hosted by multiple servers. Rules, on the other hand, are resources implementing autonomous decision logic based on a condition–action paradigm. A rule is evaluated according to the property values of selected resources, and when its condition evaluates true, it triggers specific update actions that modify the state of associated scene collections.

5.5.5. Security & privacy

IoTivity implements security mechanisms based on OCF specifications, which adopt well-established IETF standards [41]. Its architecture incorporates end-to-end protection through a layered security model that enforces authentication, authorization, and encryption across device interactions. IoTivity uses the DTLS protocol over CoAP to ensure message confidentiality and integrity in constrained environments.

Certificate-based authentication, role-based access control, and secure resource provisioning are central to the trust management framework. These mechanisms enable mutual verification between devices and services, thereby reducing the risk of unauthorized access or device impersonation. IoTivity also provides secure onboarding and credential management, enabling the dynamic enrollment of new devices while maintaining cryptographic guarantees.

From a privacy standpoint, IoTivity supports data minimization by exposing only the necessary resources and providing secure communication channels. This limits the dissemination of unnecessary data within the network. Its combination of IETF-derived standards and OCF's reference security model provide a robust foundation for trustworthy, privacy-aware IoT interoperability.

5.5.6. Market adoption

In terms of *market adoption*, IoTivity has achieved substantial recognition within the industrial and research communities due to its compliance with OCF specifications, which provide an open, standardized approach to device interoperability. Its adherence to these standards has established IoTivity as a model for evaluating cross-platform communication and standardization in IoT ecosystems.

Beyond academic use, IoTivity is integrated into several commercial and open-source initiatives that aim to promote interoperability across heterogeneous smart environments. Its open-source, modular architecture encourages experimentation, making it a popular choice among researchers, developers, and system integrators for prototyping and testing standard-compliant IoT solutions.

Although IoTivity's adoption in consumer-grade products is more limited than that of proprietary ecosystems, it remains a key enabler of standard-driven IoT development. It continues to influence the broader landscape of interoperable smart device frameworks.

5.6. DomoNet

DomoNet [42] is an open-source domotic interoperability framework created in 2006 at the Institute of Information Science and Technologies of the National Research Council of Italy (ISTI-CNR). DomoNet treats a home (and buildings in general) as an Internet node that can share environments and device functionalities with any smart objects, IoT ecosystems and other DomoNet instances throughout the Internet. For example, it can share functions and device states between the user's main residence and a vacation home to address real IoT scenarios.

5.6.1. Architecture

To this end, DomoNet (see Fig. 11) provides interfaces using Web services and SOA technologies [43] in conjunction with the expressly formulated XML-based language called DomoML. Through this approach, DomoNet also communicates with other software entities named DomoNetClients. A DomoNetClient may be a GUI that can drive DomoNet devices, or special software that receives notifications of events and, by processing them, can react by sending commands to DomoNet devices. It is possible to implement intelligent functionalities and services written using high-level abstractions of underlying domotic technologies. One example of this is DomoPredict, a client component of DomoNet able to learn users' habits by monitoring their behaviors and acting for them to actualize the so-called Ambient Intelligence [44].

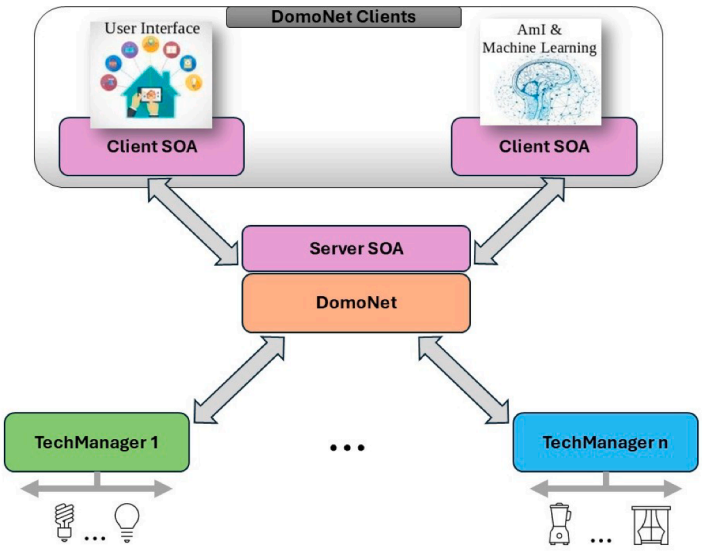

Fig. 11. DomoNet architecture.

5.6.2. Communication protocols

DomoNet supports wired and wirless communications. It implements the application and transport layers. Data exchange is disjoint because device events are not shared with the others. *DomoML* is made up of two sub-languages:

- *DomoDevice*: It Describes the characteristics of the devices, the available functions (services), the processes through which interactions with other DomoDevices must take place, and standardized data type models providing a suitable intermediate representation to enable data marshaling among heterogeneous technologies.
- *DomoMessage*: It standardizes the messages exchanged throughout the framework, which can be of two different types: (i) command: when it requests execution of a service belonging to a DomoDevice; (ii) event: when there is a status change in a DomoDevice.

In DomoNet, interaction among devices and services is defined by structured descriptions that specify communication and event response. Each device, referred to as a DomoDevice, can include special tags such as *linkedService* and *linkedInput*, which define logical associations between services and input events. With these tags, a device can automatically perform specific operations on other devices or services when certain conditions are met. For instance, when a DomoDevice executes a service such as setPower, it can trigger a corresponding web service operation–e.g., one exposed by http://www.otherwebservice.it/service with a defined identifier. This mechanism allows DomoNet to support distributed interaction patterns and coordinated control of heterogeneous smart devices without direct user intervention.

DomoNet assigns the new state value to a new power value through the linkedInput tag. This enables the sharing of values between different home automation systems, taking advantage of DomoNet's implemented data type abstraction layer. Each TechManager maps the data types of its home automation system to DomoNet data types, and vice versa, to convert values.

5.6.3. Standards & framework

In terms of Standards and Frameworks, DomoNet is an open-source middleware framework designed to abstract device heterogeneity through a modular, service-oriented architecture. It promotes interoperability across diverse smart home ecosystems by defining standardized syntactic representations for devices and services. Each device is encapsulated as a DomoDevice and exposes its functionalities through well-defined service descriptions. This enables discovery, composition, and control within the middleware. While its representations are primarily syntactic rather than semantic, DomoNet provides a solid structural basis for integrating diverse technologies and vendors into a unified operational model.

5.6.4. Integration platforms

DomoNet implements a network infrastructure to interconnect domotic devices belonging to different home automation systems. To this end, each home automation system is represented as a sub-network connected to the core of the middleware through dedicated modules called TechManagers (e.g., X10Manager, UPnPManager, KnxManager, and so on). TechManagers work as gateways between the home automation system and the DomoNet core domain logic.

The main task of DomoNet is to correctly route messages between TechManagers. In this way, devices can send and receive commands and notifications from/to other domotic networks. Moreover, DomoNet uses a comprehensive abstraction of heterogeneous home automation systems, called DomoML, to represent devices, services, interactions and events. DomoML is akin to an XML dialect and acts as the common language for domotic interoperability.

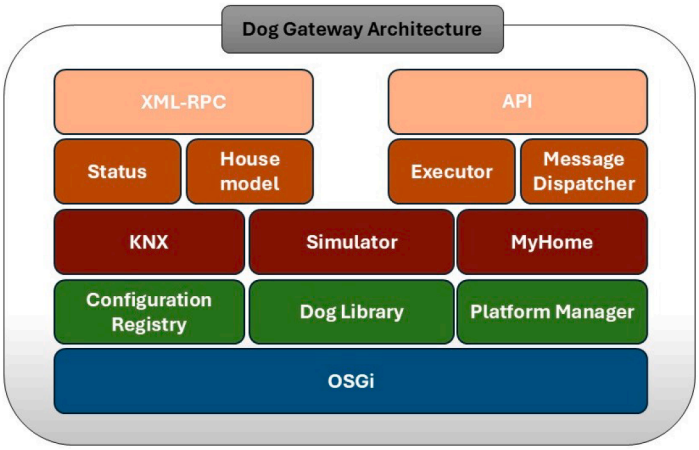

Fig. 12. Dog gateway architecture.

Being DomoNet oriented to be a network-based framework application, it can be installed in a full cloud system or in a local environment and eventually integrate third-party cloud services.

5.6.5. *Security & privacy*

In terms of Security and Privacy, DomoNet currently lacks native mechanisms for authentication, encryption, and data protection. However, its modular, service-oriented architecture allows for the integration of external security modules and privacy-preserving extensions. This flexibility in design enables the framework to accommodate evolving requirements and standards for secure data exchange, access control, and user confidentiality. Although DomoNet's architecture is not inherently security-focused, it provides a foundation upon which robust protection layers can be systematically developed and integrated.

5.6.6. *Marked adoption*

With respect to Market Adoption, DomoNet is primarily a research-oriented prototype rather than a mature commercial platform. Its open-source availability makes it accessible to developers and researchers interested in experimenting with semantic and ontology-based interoperability models. However, the framework has seen limited adoption in production or consumer environments, reflecting its role as a proof-of-concept platform that explores architectural and semantic integration principles rather than mass-market deployment.

5.7. *Dog gateway*

Dog Gateway [45] is an open-source domotic interoperability framework developed in 2006 by the Polytechnic of Turin (Italy) for research purposes. The aim of Dog Gateway is to provide HA technology able to support advanced interactions within home environments. From 2007 to 2014 Dog Gateway has been improved by adding new functionalities, integrating new home building and industrial automation technologies and providing a consistent level of device abstraction based on *DogOnt* [46], a well-known ontology in the domain of domotics whose purpose is to structurally describe smart environments. As storage layer Dog Gateway uses the Jena framework [47], including a triplestore specifically designed to handle RDF graphs and ontologies.

5.7.1. *Architecture*

Dog Gateway is organized in 4 layers (see Fig. 12) covering from low-level interconnection to high-level modeling and interfacing.

- *Layer 0*: It implements the control and management of interactions between the OSGi platform and the other bundles (e.g. system and runtime events, errors and failures) in the other levels.
- *Layer 1*: It interfaces HA technologies to which the framework can be connected using specific drivers.
- *Layer 2*: It implements routing to address messages and implements the DogOnt ontology.
- *Layer 3*: It permits access to external applications.

5.7.2. *Communication protocols*

In terms of communication protocols, the Dog Gateway supports wired and wireless communication media. This enables integration with a wide variety of home automation devices and subsystems. It implements the *application* and *transport* layers to ensure reliable device interaction and protocol abstraction. Data exchange follows a disjoint model, meaning communication between different types of devices is mediated through the gateway rather than through direct links between devices.

The platform has a modular, adapter-based architecture that enables the integration of various communication technologies, including ZigBee, Z-Wave, KNX, and Modbus, without altering the core logic. Each device type is represented as a service within the middleware, and protocol-specific drivers handle message translation and event propagation. This abstraction layer simplifies interoperability across heterogeneous networks and facilitates extensibility, allowing developers to introduce new protocols through plug-in components.

5.7.3. Standards & frameworks

Dog Gateway is an open-source framework with middleware functions. It is built on a layered architecture compliant with OSGi standards, enabling core internal communication and event management. The system includes a component called the House Model, which maintains and updates a semantically structured representation of the environment. This component provides access to device statuses, supported operations, functionalities, and data types. The House Model serves as a centralized knowledge hub that other system components can query to retrieve information about all devices, their contexts, and capabilities. To represent objects, it uses a semantic approach.

5.7.4. Integration platforms

Dog Gateway uses specialized smart gateways acting as drivers to translate messages between Dog Gateway and legacy home automation networks, and vice versa. It uses the *DogMessage* formalism, which is independent of the technologies used in the platform to exchange messages inside the entire framework. A *MessageDispatcher* software component addresses messages to the proper network drivers.

To implement interoperability, Dog Gateway utilizes a Rule Engine based on the Drools DRL language [48]. When Rule Engine verifies a conditional statement, it executes the corresponding command

Dog Gateway is not cloud-based and can be installed locally.

5.7.5. Security & privacy

With respect to security and privacy, the Dog Gateway was conceived primarily as a research platform for experimenting with interoperability and middleware integration, rather than as a production-grade system. Consequently, security and privacy mechanisms were not central to its initial design. The framework does not natively implement features such as encryption, authentication, or access control, instead of relying on underlying communication protocols or network configuration for basic protection.

Nevertheless, its modular architecture allows for the integration of additional security layers, such as encrypted communication channels or identity management modules, for specific deployments as needed. This design flexibility makes it a suitable foundation for academic or experimental extensions that explore privacy-preserving interoperability or secure middleware architectures.

5.7.6. Market adoption

Concerning market adoption, Dog Gateway is primarily a research prototype developed in academic and experimental contexts. While it effectively demonstrates middleware-based interoperability concepts, it has not undergone the engineering, optimization, and validation processes necessary for large-scale or commercial deployment. Consequently, it is currently only adopted by research institutions and specialized developers interested in exploring interoperability frameworks. However, its modular architecture provides a solid foundation for its potential future evolution into more deployable, standardized solutions.

5.8. Comparative results

Table 2 provides a concise comparison of the HA frameworks introduced and analyzed previously, based on the taxonomy entries. A ✓indicates that the framework supports the feature, an empty cell indicates that it does not, and a △ indicates that the framework only partially supports it. In evaluating the table entries, it is essential to consider both syntax and semantics in object representation. As previously discussed, syntax refers to the structures and formats used to describe the actors of the system, such as devices, attributes, features, states and events, leaving the interpretation of their meaning to the user. Within an ontological paradigm, semantics specifies the meaning of these entities, formally defining how the system must process, interpret, and use them. The taxonomy-based analysis of the frameworks reveals several recurring patterns. Most platforms prioritize local, modular integration and adopt rule-based automation, often leveraging community-developed plugins. While syntactic interoperability is generally well supported, semantic interoperability and formal security mechanisms remain limited. The comparison also reveals varying degrees of openness and extensibility. These frameworks appeal to technically skilled users, but many lack the usability features necessary for broader, mainstream adoption. These findings underscore the necessity of unified, user-friendly, semantically enriched platforms that can address the evolving demands of smart home environments.

Overall, when analyzed through the lens of the proposed interoperability taxonomy, several convergence patterns and gaps emerge among the examined frameworks. Although developed independently, these frameworks converge around a shared operational paradigm: they enable rule-based automation across devices and services by abstracting protocol heterogeneity. Most platforms operate on affordable local hardware with optional cloud extensions, a design that enhances privacy and reliability through local control.

Despite these commonalities, notable limitations persist across all solutions. Security mechanisms are generally limited to basic authentication and encryption, offering only partial protection against emerging threats. Semantic interoperability also remains underdeveloped–only one framework employs metadata to represent object roles, while others rely on purely structural descriptions.

Table 2
Comparison of home automation frameworks via taxonomy (an empty cell means no, ✓means yes, and △ means partially supported).

Framework	Communication Protocols											Standards and Frameworks				Integration Platforms						Security and Privacy			Market Adoption		
	Comm. Medium		Comm. Layer			Data Exchange		Standards		Frameworks			Object Repr.			Hub-based		Cloud-based		Local Integr.							
	Wireless	Wired	App.	Transp.	Hybrid	Shared	Disjoint	OpenSrc	Propr.	OpenSrc	Propr.	Middleware	Synt.	Sem.		Single	Multi	Full	Hybrid	Standalone	D2D	Encrypt.	Cert.	Privacy	Wide	Niche	Restr.
OpenHAB	✓	✓	✓	✓			✓	✓	✓	✓		✓	✓	✓			✓		✓	✓		△	△	✓		✓	
ioBroker	✓	✓	✓	✓		✓	△	✓	✓	✓		✓	✓	✓			✓	△	✓	✓		△	△	△			✓
Domoticz	✓	✓	✓	✓			✓	✓	✓	✓			✓	✓			✓	✓	✓	✓		✓	✓	△	✓		
Home Assistant	✓	✓	✓	✓			✓	✓	✓	✓		✓	✓				✓		✓	✓		△	△	✓	✓		
IoTivity	✓	✓	✓	✓		✓	✓	✓		✓		✓	✓				✓		✓	✓	✓	✓	✓	✓			✓
DomoNet	✓	✓	✓	✓			✓	✓		✓		✓	✓	△			✓	✓	✓	✓							✓
Dog Gateway	✓	✓	✓	✓			✓	✓		✓		✓		✓			✓		✓	✓							✓

Most adopt a device-centric abstraction model organized by type and capability, typically accessed through user-friendly interfaces and mobile applications.

For comparison, DomoNet offers a conceptually similar, though less mature, approach with limited market adoption. Initially centered on syntactic device integration, it has since incorporated a semantic layer based on ontologies and extended its functionality to support advanced features such as IPv6 connectivity through external APIs [49].

Taken together, these findings reveal a convergence toward lightweight, rule-based architectures that prioritize usability and local operation, while exposing systemic gaps in semantic reasoning, adaptive behavior, and robust security, key areas that must be addressed to achieve seamless interoperability in future HA ecosystems.

Considering the examined middleware platforms as representative of the domain, it is evident that several substantial research and implementation challenges remain.

The first key challenge is adopting semantic technologies that can unambiguously contextualize the environment and the data collected within it. Semantic information would allow systems to correctly interpret, understand, and process events in the user's surroundings. Although there are preliminary efforts in Dog Gateway and DomoNet, semantic modeling is only partially implemented across frameworks.

A related, more ambitious challenge is achieving real-time system responsiveness. Once semantic context is available, artificial intelligence algorithms could interpret user needs and environmental conditions autonomously, reducing reliance on predefined, rule-based automations. However, none of the analyzed platforms currently support this form of adaptive, AI-driven decision-making.

User-system interaction remains an open issue with significant practical implications. While most systems rely on web interfaces, only Domoticz and Home Assistant offer dedicated Android and iOS applications. Voice-based interfaces supported by natural language understanding are a promising way to improve accessibility and usability.

From an architectural standpoint, the analyzed frameworks depend heavily on standard cybersecurity mechanisms, such as OAuth 2.0 authentication and HTTPS communication. However, they do not incorporate domain-specific protections. IoTivity partially addresses this gap by implementing advanced OCF security specifications. In terms of privacy, while all platforms support local operation without external cloud dependencies, many lack comprehensive privacy-by-design mechanisms.

As IoT ecosystems grow, the need for distributed architectures becomes more relevant. An emerging requirement is deploying multiple coordinated instances of the same platform across different locations while ensuring seamless collaboration. Of the systems surveyed, only DomoNet and ioBroker natively support this capability. Another closely related requirement is the ability to securely expose devices to external networks, enabling transparent, standards-based access without relying on proprietary middleware. DomoNet explores this paradigm through public IPv6 addressing and REST-based interfaces. Data modeling practices also differ. While most systems rely on centralized databases for harmonization, DomoNet models and shares data directly with connected devices internally, thus avoiding the need for a centralized repository.

From a technical perspective, most platforms have not yet been fully engineered for large-scale deployment and often require specialized expertise. Domoticz and Home Assistant are partial exceptions due to their intuitive interfaces and active user communities.

Finally, a particularly difficult challenge is developing a true plug-and-play platform that can integrate any device or home automation standard without relying on additional software modules.

6. Conclusion

In this study, we reviewed two decades of progress in Home Automation (HA), focusing on interoperability frameworks. Although recent initiatives, such as *Matter*, have addressed longstanding challenges, the field still lacks a fully unified, widely adopted interoperability model.

To address this gap, we introduced a structured taxonomy specifically designed to analyze and compare interoperability solutions in HA. When applied to several prominent frameworks, the taxonomy clarifies key characteristics, reveals recurring patterns, and exposes limitations in current systems.

Although meaningful advances have been made, our analysis shows that full interoperability is still restricted by limited semantic reasoning, inadequate adaptive behavior, and inconsistent security practices. The proposed taxonomy provides a systematic approach to evaluating and categorizing these issues, revealing where current methods align and where they fall short. These findings will guide future research, emphasizing the necessity of more intelligent, secure, and context-aware interoperability solutions. Ultimately, the taxonomy establishes a common language for evaluating HA frameworks, supporting transparent design decisions and coherent development strategies for interoperable smart home ecosystems.

Overall, the proposed taxonomy proved effective in revealing convergence patterns, distinctive design choices, and persistent gaps across current HA frameworks, offering a clear foundation for evaluating interoperability in heterogeneous smart home ecosystems. However, our analysis shows that most existing solutions focus primarily on syntactic interoperability and rule-based control. These solutions offer limited support for semantic reasoning, adaptive behavior, and robust security and privacy guarantees. Open-source frameworks are increasingly adopting hybrid cloud-local architectures and critical protections, such as encryption, standardized security models, and privacy-by-design mechanisms; however, these protections are implemented inconsistently. These findings underscore the ongoing misalignment between current technical capabilities and user expectations for secure, context-aware, and interoperable home environments.

Looking ahead, several research directions emerge. Semantic technologies could significantly improve device integration and automation, while interoperability models must increasingly incorporate built-in security and privacy mechanisms as smart homes rely more on cloud services and third-party ecosystems. Additionally, the taxonomy could be expanded to include dimensions such

as user experience, energy efficiency, and emerging contexts, including multi-vendor environments, hybrid edge-cloud deployments, and AI-driven smart home solutions. Future work will focus on validating the taxonomy through expert feedback and developing practical guidelines and tools to support its adoption. Incorporating trust, data sovereignty, and compliance considerations into future extensions will further enhance its ability to capture the evolving landscape of HA interoperability.

CRediT authorship contribution statement

Dario Russo: Methodology, Investigation, Conceptualization; **Vittorio Miori:** Supervision, Funding acquisition; **Gabriele Tolomei:** Writing – review & editing, Supervision; **Dimitri Belli:** Writing – original draft, Methodology, Investigation, Conceptualization.

Data availability

Data will be made available on request.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the Age-It project, funded by the European Union – Next Generation EU, within the framework of the National Recovery and Resilience Plan, under the Extended Partnership Initiative PE8 ‘Conseguenze e sfide dell’invecchiamento’, Project Age-It (CUP: B83C22004880006) .

References

[1] M. Ilchenko, L. Uryvsky, L. Globa, Progress in Advanced Information and Communication Technology and Systems, Cham: Springer International Publishing (548) (2023).
[2] S.T. Arzo, C. Naiga, F. Granelli, R. Bassoli, M. Devetsikiotis, F.H.P. Fitzek, A theoretical discussion and survey of network automation for IoT: challenges and opportunity, IEEE IoT 8 (15) (2021) 12021–12045.
[3] M.R. Mahmood, M.A. Matin, P. Sarigiannidis, S.K. Goudos, A comprehensive review on artificial intelligence/machine learning algorithms for empowering the future IoT toward 6G era, IEEE Access 10 (2022) 87535–87562.
[4] G. Paolone, D. Iachetti, R. Paesani, F. Pilotti, M. Marinelli, P. Di Felice, A holistic overview of the Internet of Things ecosystem, IoT 3 (4) (2022) 398–434.
[5] M. Lombardi, F. Pascale, D. Santaniello, Internet of Things: a general overview between architectures, protocols and applications, Information 12 (2) (2021).
[6] E. Al-Masri, K.R. Kalyanam, J. Batts, J. Kim, S. Singh, T. Vo, C. Yan, Investigating messaging protocols for the Internet of Things (IoT), IEEE Access 8 (2020) 94880–94911.
[7] V.A. Orfanos, S.D. Kaminaris, P. Papageorgas, D. Piromalis, D. Kandris, A comprehensive review of IoT networking technologies for smart home automation applications, J. Sensor Actuat. Netw. 12 (2) (2023) 30.
[8] P. Barsocchi, D. Belli, E. Gabrielli, D.L. Rosa, V. Miori, F. Palumbo, D. Russo, G. Tolomei, A novel architectural schema for constant monitoring and assessment of older adults’ health status at home, in: International Conference on Pervasive Computing Technologies for Healthcare, Springer, 2023, pp. 501–511.
[9] M. Kumar, R.K. Gupta, G. Jain, P. sharma, Next-Generation smart homes: an IoT approach to home automation and user experience, in: 2024 International Conference on Artificial Intelligence and Quantum Computation-Based Sensor Application (ICAIQSA), 2024, pp. 1–6.
[10] L. Martirano, M. Mitolo, Building automation and control systems (bacs): a review, in: 2020 IEEE International Conference on Environment and Electrical Engineering and 2020 IEEE Industrial and Commercial Power Systems Europe (EEEIC/I&CPS Europe), IEEE, 2020, pp. 1–8.
[11] G. Kayas, M. Hossain, J. Payton, S.M.R. Islam, An overview of UPnP-based IoT security: threats, vulnerabilities, and prospective solutions, in: 2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), IEEE, 2020, pp. 0452–0460.
[12] A.A. Zaidan, B.B. Zaidan, A review on intelligent process for smart home applications based on IoT: coherent taxonomy, motivation, open challenges, and recommendations, Artif. Intell. Rev. 53 (1) (2020) 141–165.
[13] J.S. Edu, J.M. Such, G. Suarez-Tangil, Smart home personal assistants: a security and privacy review, ACM Comput. Surv. 53 (6) (2020).
[14] D. Belli, P. Barsocchi, F. Palumbo, Connectivity standards alliance matter: state of the art and opportunities, IoT (2023) 101005.
[15] M. Kasunic, W. Anderson, Measuring systems interoperability: challenges and opportunities (2004).
[16] T. Clark, R. Jones, Organisational interoperability maturity model for C2, in: Proceedings of the 1999 Command and Control Research and Technology Symposium, 29, Citeseer, 1999, pp. 1–13.
[17] A. Tolk, J.A. Muguira, The levels of conceptual interoperability model, in: Proceedings of the 2003 Fall Simulation Interoperability Workshop, 7, Citeseer, 2003, pp. 1–11.
[18] A. Tolk, S.Y. Diallo, Model-based data engineering for web services, IEEE Internet Comput. 9 (4) (2005) 65–70.
[19] A. Tolk, S.Y. Diallo, C.D. Turnitsa, Applying the levels of conceptual interoperability model in support of integratability, interoperability, and composability for system-of-systems engineering, J. Syst. Cybernet. Informat. 5 (5) (2007).
[20] N. Ide, J. Pustejovsky, What does interoperability mean, anyway? Toward an operational definition of interoperability for language technology, in: Proceedings of the Second International Conference on Global Interoperability for Language Resources. Hong Kong, China, 2010, pp. 1–8.
[21] A.A. Salatino, T. Thanapalasingam, A. Mannocci, F. Osborne, E. Motta, The computer science ontology: a large-scale taxonomy of research areas, in: International Semantic Web Conference, Springer, 2018, pp. 187–205.
[22] O. Taiwo, L.A. Gabralla, A.E. Ezugwu, Smart home automation: taxonomy, composition, challenges and future direction, in: International Conference on Computational Science and Its Applications, Springer, 2020, pp. 878–894.
[23] L. Andraschko, P. Wunderlich, D. Veit, S. Sarker, Towards a taxonomy of smart home technology: a preliminary understanding, 2021.
[24] J.F. DeFranco, M. Kassab, Smart home research themes: an analysis and taxonomy, Proced. Comput. Sci. 185 (2021) 91–100. Big Data, IoT, and AI for a Smarter Future.
[25] S. FakhrHosseini, C. Lee, S.-H. Lee, J. Coughlin, A taxonomy of home automation: expert perspectives on the future of smarter homes, Inform. Syst. Front. 27 (2) (2025) 449–466.

[26] T. Mzili, M. Mzili, S.I. Bouderba, A. Abatal, W. Aribowo, A.K. Arya, Interoperability in Internet of Things: taxonomies and open challenges, Babyl. J. IoT 2025 (2025) 101-112.
[27] A.E. Ezugwu, O. Taiwo, O.S. Egwuche, L. Abualigah, A. Van Der Merwe, J. Pal, A.K. Saha, A.I. Alzahrani, F. Alblehai, J. Greeff, M.O. Olusanya, Smart homes of the future, Transact. Emerg. Telecommun. Technolog. 36 (1) (2025).
[28] A. Kujur, Z. Raza, A.A. Khan, C. Wechtaisong, Data complexity based evaluation of the model dependence of brain MRI images for classification of brain tumor and Alzheimer's disease, IEEE Access 10 (2022) 112117–112133.
[29] A.A. Khan, M. Faheem, R.N. Bashir, C. Wechtaisong, M.Z. Abbas, Internet of things (IoT) assisted context aware fertilizer recommendation, IEEE Access 10 (2022) 129505–129519.
[30] R.N. Bashir, F.A. Khan, A.A. Khan, M. Tausif, M.Z. Abbas, M.M.A. Shahid, N. Khan, Intelligent optimization of reference evapotranspiration (ETo) for precision irrigation, J. Comput. Sci. 69 (2023) 102025.
[31] A.A. Khan, M.A. Nauman, R.N. Bashir, R. Jahangir, R. Alroobaea, A. Binmahfoudh, M. Alsafyani, C. Wechtaisong, Context aware evapotranspiration (ETs) for saline soils reclamation, IEEE Access 10 (2022) 110050–110063.
[32] A.A. Khan, M. Driss, W. Boulila, G.A. Sampedro, S. Abbas, C. Wechtaisong, Privacy preserved and decentralized smartphone recommendation system, IEEE Trans. Consum. Electron. 70 (1) (2023) 4617–4624.
[33] B.K. Sovacool, D.D. Furszyfer Del Rio, Smart home technologies in Europe: a critical review of concepts, benefits, risks and policies, Renew. Sustain. Energy Rev. 120 (2020) 109663.
[34] R.C. Nickerson, U. Varshney, J. Muntermann, A method for taxonomy development and its application in information systems, Eur. J. Inform. Syst. 22 (3) (2013) 336–359.
[35] V. Miori, D. Russo, M. Aliberti, Domotic technologies incompatibility becomes user transparent, Commun. ACM 53 (1) (2010) 153–157.
[36] A. Deohate, D. Rojatkar, Middleware challenges and platform for IoT-A survey, in: 2021 5th International Conference on Trends in Electronics and Informatics (ICOEI), 2021, pp. 463–467.
[37] B. Setz, S. Graef, D. Ivanova, A. Tiessen, M. Aiello, A comparison of open-source home automation systems, IEEE Access 9 (2021) 167332–167352.
[38] A. Prakash, S.I. Basha, OSGi Basics, in: Hands-On Liferay DXP: Learn Portlet Development and Customization Using OSGi Modules, Springer, 2022, pp. 1–22.
[39] A. Nierbeck, J. Goodyear, J. Edstrom, H. Kesler, Apache Karaf Cookbook, Packt Publishing Ltd, 2014.
[40] S. Park, OCF: A new open IoT consortium, in: 2017 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA), IEEE, 2017, pp. 356–359.
[41] A. Gharib, J. Jaskolka, M. Ibnkahla, A. Matrawy, Security management of horizontal IoT platforms: a survey and comparison, ACM Comput. Surv. 58 (3) (2025). https://doi.org/10.1145/3766888
[42] V. Miori, D. Russo, C. Concordia, Meeting people's needs in a fully interoperable domotic environment, Sensors 12 (6) (2012) 6802–6824.
[43] L. Haorongbam, R. Nagpal, R. Sehgal, Service oriented architecture(SOA): a literature review on the maintainability, approaches and design process, in: 2022 12th International Conference on Cloud Computing, Data Science and Engineering (Confluence), 2022, pp. 647–652. https://doi.org/10.1109/Confluence52989.2022.9734153
[44] G.-A. Hernández-Torres, E. Sánchez-DelaCruz, Challenges and opportunities of ambient intelligence (AmI) in the 21st century: a historical review, Evol. Intell. 18 (4) (2025) 80. https://doi.org/10.1007/s12065-025-01067-1
[45] D. Bonino, E. Castellina, F. Corno, The DOG gateway: enabling ontology-based intelligent domotic environments, IEEE Trans. Consum. Electron. 54 (4) (2008) 1656–1664.
[46] D. Bonino, L. De Russis, DogOnt as a viable seed for semantic modeling of AEC/FM, Semant. Web. 9 (6) (2018) 763–780.
[47] A. Krygin, P. Karpenko, O. Sychev, Performing first-order-logic queries over RDF data: interpreter versus compiler to apache Jena rules, in: P. Mylonas, D. Kardaras, J. Caro (Eds.), Novel and Intelligent Digital Systems: Proceedings of the 4th International Conference (NiDS 2024), Springer Nature Switzerland, Cham, 2024, pp. 537–548.
[48] P. Browne, P. Johnson, JBoss Drools business rules, 1847196063, Packt Publishing Birmingham, 2009.
[49] V. Miori, D. Russo, Anticipating health hazards through an ontology-based, IoT domotic environment, in: 2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, IEEE, 2012, pp. 745–750.