The HESTIA Framework

From an Internet of Things to an Internet of Meaning

Marianne Schnellmann and Henderik A. Proper

Abstract At first glance, the Internet of Things brings about an expectation for users (be it individuals or organizations) to interact with the many Internet-connected "things" in a natural way while also enhancing everyday work and life. The emergence of smart cities, and smart homes, also fuels the need for a broad audience to interact with the Internet of Things in a natural way. In current practice, however, users are confronted with the need to negotiate a complex landscape involving a myriad of protocols, standards, and work-arounds to integrate "legacy" devices, etc. We contend that users should not have to think about their world in terms of specific sensors, actuators, gateways, and protocols but rather in terms of room temperatures, the desire to increase the temperature in the living room, the concern that the plants in the garden are watered on time, etc. This creates a need to bridge this gap by creating a semantically meaningful layer of abstraction on top of the sensors and actuators that make up the "device and protocols oriented" Internet of Things, to create an Internet of Meaning. To this end, this chapter reports on the HESTIA framework, which combines: (1) An abstraction of the implementation details pertaining to, e.g., different protocols, standards, etc. (2) A domain-specific (conceptual) modeling framework in terms of which "things" can be captured in a way that is meaningful to the domain at hand (3) Based on this, a domain-specific language that is understandable by the user, enabling users to define control/behavioral rules in terms that are meaningful to them The presented HESTIA framework will be illustrated in terms of examples in the context of home and garden automation. Though such application contexts seem less challenging and complex than industrial Internet of Things applications, the variety of devices and protocols and distance between users and the technical details are often larger than in the case of industrial Internet of Things.

Keywords Internet of Meaning · HESTIA · Domain-Specific Modeling

M. Schnellmann (✉) · H. A. Proper
TU Wien, Vienna, Austria
e-mail: marianne.schnellmann@tuwien.ac.at; henderik.proper@tuwien.ac.at

X. Boucher et al. (eds.), *Domain-Specific Conceptual Modeling*,
https://doi.org/10.1007/978-3-031-98660-4_11

1 Introduction

As part of the general urbanization trend, more and more people move into cities. According to the United Nations [1], by 2050, 68% of the world's population will actually be living in cities. Hand in hand with the growth of the population of cities, several challenges arise in the areas of housing, transport, and sustainable resource management [2]. To help cope with these challenges, modern-day digital technologies are integrated into the infrastructures of cities, including houses, turning them into smart cities and smart homes, respectively [2].

Smart homes are often seen as being the smallest unit of a smart city [3]. They allow manual processes "around the house" to be digitalized, automated, and coordinated. We can also observe how household appliances increasingly become smart appliances, helping residents with everyday tasks [4].

More generally, the core of smart cities and smart homes is shaped by advanced integrated systems that combine physical processes with digital control mechanisms [5], which are present in various technical applications [6–8]. The Internet of Things (IoT) supplies the fundamental infrastructure for these systems [9]. Due to the numerous interdependencies among individual components, developing IoT-based systems is highly complex [6]. This increasing complexity is a key challenge when designing and developing [6]. For instance, in the context of smart homes, this is exemplified by the fact that there is no unified standard for smart home devices. Instead, many different protocols and systems compete with each other, which in turn leads to an increased complexity [3, p. 5]. For instance, there are presently 600 IoT platforms on the market [10].

This complexity also confronts the (generally non IT experts) users of IoT. Even though, at first glance, the Internet of Things brings about the expectation for users (be it individuals or organizations) to interact with Internet-connected "things" in a natural way, reality is far from it. Taking into account the interaction with the user and their limited technical skills, setting up and maintaining, e.g. a smart home system is often complex and difficult to accomplish [11]. As such, current practice still confronts users with a need to negotiate a complex landscape, involving a myriad of protocols, standards, and technical work-arounds (including dealing with "legacy" devices and other technical constraints). In addition, we contend that users do not (want/need to) think about their world in terms of sensors and actuators. They do, however (want to), think in terms of room temperatures, energy consumption, the desire to increase the temperature in the living room, making sure that the plants in the garden are watered on time, etc.

This leads to a need for ways to deal with such complex systems and to understand and structure them in more natural ways. This fuels the need to bridge this gap by the creation of a semantically meaningful layer of abstraction on top of the sensors and actuators that make up the "device-oriented" Internet of Things, to create an Internet of "Things with Meaning."

In creating such a semantic layer, we make use of the concept of conceptual models. A conceptual model [12–14] allows one to identify the essential elements of

a domain, as well as capture the relations between them and visualize dependencies. This enables us to focus on the important aspects of the problem while abstracting from the rest [12–14]. Conceptual models also provide a common language in terms of a visual representation. By using them, complexity can be reduced, relationships can be presented more effectively, and misunderstandings can be minimized [15].

To this end, this chapter reports on the HESTIA framework, which combines:

1. An abstraction of the implementation details pertaining to, e.g., different protocols, standards, and technical work-arounds
2. A domain-specific (conceptual) modeling framework in terms of which "things" can be captured in a way that is meaningful to the domain at hand
3. Based on this, a domain-specific language that is understandable by the user, enabling users to define control/behavioral rules in terms that are meaningful to them.

The HESTIA framework will be illustrated in terms of examples in the context of home and garden automation. Though such application contexts seem less challenging and complex than industrial Internet of Things applications, the variety of devices and protocols and distance between users and the technical details are often actually larger than in the case of industrial Internet of Things. Our personal experiences with Open Source platforms such as openHAB,[1] FHEM,[2] and HomeAssistant[3] illustrate this point quite vividly.

The remainder of this chapter is structured as follows. In Sect. 2, the general architecture of IoT systems as used in HESTIA is presented. Based on this, the operationalization of this architecture toward an actual platform is discussed in Sect. 3.1. A model conceptualization, and illustration, of the HESTIA framework and platform is then provided in terms of a domain-specific language for smart gardens (Sect. 3.2) and a small case study of a (miniature) garden (Sect. 4). Before concluding, Sect. 5 then provides a brief reflection of the latter proof of concept while also identifying opportunities for further research.

2 Method Description: General Architecture

The overall architecture for IoT, as used in the HESTIA framework, is depicted in Fig. 1, in terms of a UML [16] class diagram. The main aspects involved in this architecture are discussed below.

[1] https://www.openHAB.org.

[2] https://fhem.de.

[3] https://www.home-assistant.io.

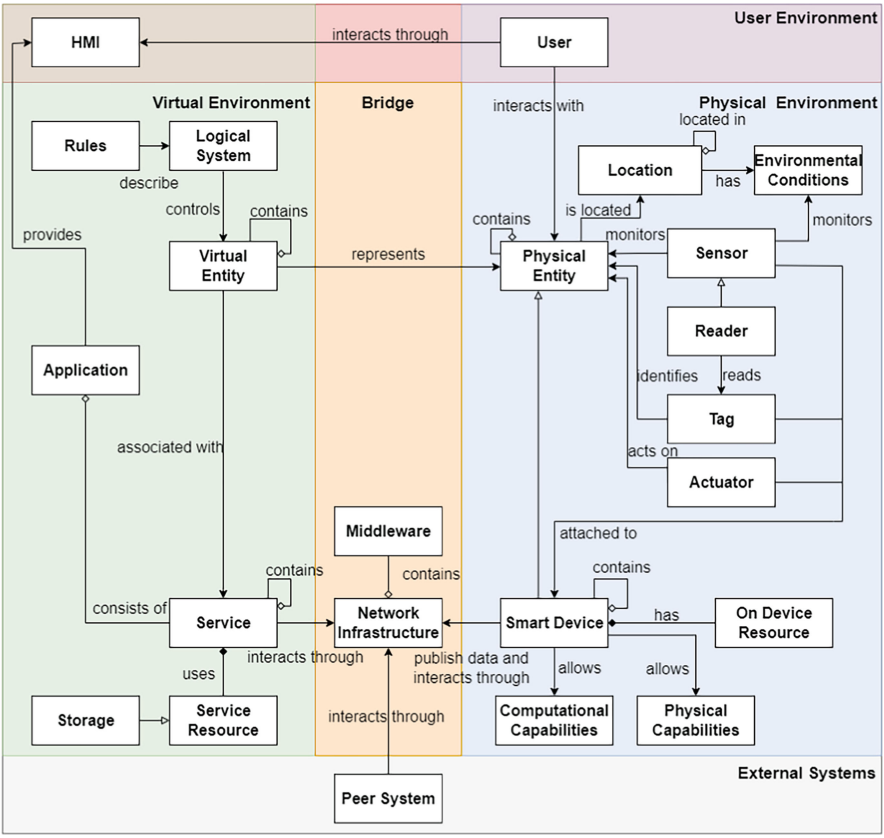


Fig. 1 General architecture of IoT-based systems for HESTIA

2.1 Physical Environment

The *physical environment* parts of the architecture are concerned with the physical manifestation of the world around us. It already intends to provide an abstraction of the IoT but rather in terms of the physical entities one may observe. As such, a *Physical Entity* pertains to a discrete, identifiable, and observable object [17, 18] in the real world [19], which has relevance to a user.

In principle, any object can be considered a physical entity as long as it is of interest to the user to achieve their goals [20, 21]. Examples of such objects include people, animals, household appliances, premises, vehicles, jewelry, clothing, and electronic devices [20–22]. In general, one can track, monitor, or interact with them [21]. Furthermore, a physical entity may "*contain*" several other physical entities.

Physical entities and the associated devices are located at a specific *Location* or region, which can be located using the global positioning system (GPS) [23]. The

location may also involve data, such as the address or even the building, room, and floor number [24]. At this location, there are dynamic *Environmental Conditions* such as the weather, which may influence the physical entities [23].

Physical entities may be observed by *Sensors*, reporting data about these entities [22, 25]. In terms of, e.g., sight, hearing, smell, taste, touch, and temperature, they can recognize the physical environment and the related environmental conditions [25]. This may range from identity to measurements of the state of the physical entity. The data collected by sensors can be recorded and retrieved at any time [20], to, e.g., inform *users* or *applications* about the state of the physical entities and their environment.

Tags can be used as an identifier for a physical entity. The tags are read by a special kind of sensor (specialization), which is usually referred to as *Readers*. This serves to support the identification process. This process can be optical, using bar codes and QR codes, or based on radio frequency, as in the case of microwave license plate recognition systems and RFID [20, 22].

Actuators can change the physical state of a physical entity [22]. They accept digital input and, based on this, act on the properties of one or more physical entities [26]. Examples include translating, rotating, stirring, inflating, switching on/off, or activating/deactivating functions [20, 22].

A *Smart Device* involves hardware for monitoring or interacting with environmental conditions in the real world or the physical entity. The applied specialization represents an is-a relationship. Actuators, sensors, or tags can be attached to the smart device or otherwise embedded in it. On the one hand, the specialization is disjoint, in the sense that a sensor cannot also be an actuator and vice versa [20]. On the other hand, they are partial. In other words, further actuators, sensors, or tags can also be added to the device and makes it even "smarter"; the list is not exhaustive. In the context of Home Automation, an example would be an (off-the-shelf) *physical entity* that reports the temperature (*sensor*), the humidity (*sensor*), and the air pressure (*sensor*). Another example would be a radiator thermostat that measures (and reports) the room temperature (*sensor*) and controls (*actuator*) the radiator's valve based on a set target temperature. However, such physical entities can also be located in their environment and monitor its conditions indirectly (e.g., a motion sensor) [19].

As a smart device is part of the physical environment, it counts as a physical entity itself [19]. A device can have *computational capabilities* [27], *physical capabilities*, and access *on device resources* such as local memory resources [18].

2.2 Virtual Environment

The Internet of Things (IoT) is primarily concerned with connecting the physical world with the virtual world. Physical entities are represented in the virtual or digital world by a *Virtual Entity*. While there is only one physical entity for each virtual entity, it is possible that the same physical entity is linked to several virtual entities,

e.g., due to application domain-related representations [20]. One virtual entity might represent a lamp (physical entity) as part of the overall smart home system, while another virtual entity could specify the lamp's location, such as *being in the living room*.

Each virtual entity must have a unique identifier [20]. All relevant digital parameters that represent the characteristics of the physical entity are updated whenever there is a change in the physical environment. In the same way, changes relating to the virtual entity can affect the physical entity in the real world [22].

To enable users to define control/behavioral rules in a form that makes sense for them, the concept *Rules* has been created. Several rules together form the *Logical System*, which controls the Physical Entity via the Virtual Entity according to the needs of the user.

The user accesses the system via a service client, i.e., an *application* [27] with an accessible user interface (HMI) [20]. The application consists of *services* or software components [24], which are designed to perform certain functions or tasks. Services are usually implemented as independent programs that can be used and executed independently of each other [18]. It provides either information about the physical entity or acts on it and is connected to the corresponding virtual entity [17]. The individual services use data storage *Storage* [18]. These are resources that the service uses. As a service requires not only storage resources but also security or network resources, a standardized term *Service Resource* is used to summarize all resources.

2.3 User Environment

Depending on how the IoT system is used, users vary from everyday individuals, who engage with basic system functionalities as end users, to people with specific roles (e.g., owner, administrator) and machines, devices, or services [19]. The interaction between a *user* and a physical entity can occur in two ways. On the one hand, the user interacts directly with the physical entity in the real world (e.g., by manual operation). Then, the user accesses a human-machine interface (HMI) in the virtual environment [20], which either provides information about the physical entity or influences it through user intervention.

HMI entities are therefore used to control the system. It displays the important data collected by the sensors and allows the user to change parameters of the virtual entity [28]. These HMIs are usually easy to use and use elements such as switches, buttons, sliders, and displays and often try to visualize the real environment [29].

2.4 Bridge and External Systems

Controllers (Smart Devices) such as Arduino-, Pycom-, Raspberry Pi-based systems collect data and interact with at least one *Network*. A network is an infrastructure that supports the communication or transmission of data [26] between machine to machine (M2M) with the support of communication protocols [27]. Devices can send the data to the nearest gateway using communication protocols such as Z-Wave, MQTT, HTTP, Bluetooth, Wi-Fi, or Zigbee [30]. *Middleware* includes endpoints, gateways, communication protocols, etc. This allows the devices to establish connectivity to the IoT systems [19]. A *Peer System* can be another IoT systems or a user device or offer services to the IoT system [31]. The peer system interacts with the IoT system via the user network, typically via the Internet [18, 31].

3 Method Conceptualization

3.1 The Platform

In this section, we discuss a possible operationalization of the architecture as shown in Fig. 1, in terms of a platform for the HESTIA framework as also used in the experimental setup to be reported in Sect. 4. The general architecture of the platform is shown in Fig. 2.

The *physical layer* involves the smart devices and their connectivity via the network infrastructure with the virtual environment. The *conceptual modeling layer* involves a conceptual model of the *application domain* in terms of (a domain-specific specialization of) the general architecture as shown in Fig. 1. The *scenario layer* is used for representing real-world scenarios of how people would interact with the smart home/garden system. In Sect. 4, this layer is used for evaluation purposes.

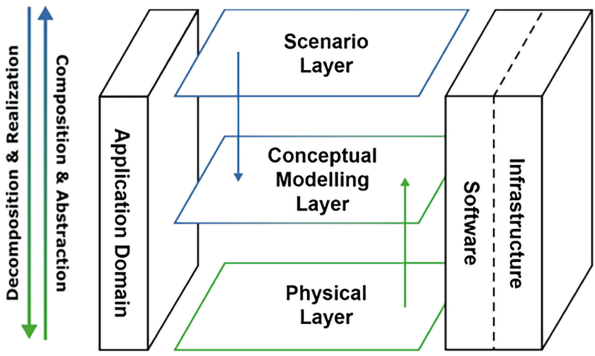


Fig. 2 Layer-based architecture of the experiment (Inspired by [32])

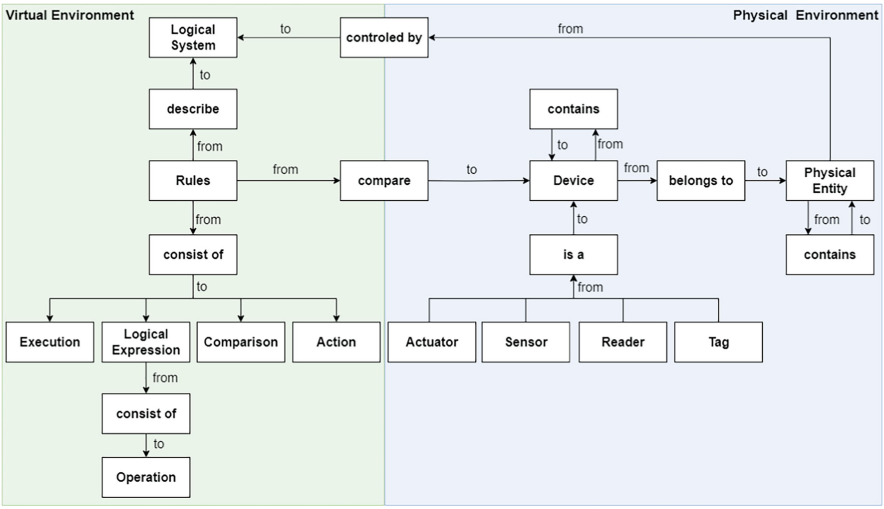


Fig. 3 HESTIA metamodel

The *application domain* actually provides input for the three layers and describes the domain of interest. As such, it also defines the domain-specific (conceptual) modeling framework in terms of which "things" can be captured in a way that is meaningful to the domain at hand at the *conceptual modeling* layer. To enable this, a more specific metamodel for the HESTIA modeling method was created, based on Fig. 1.

On the one hand, the resulting metamodel focuses on the physical environment or the inclusion of the application area. On the other hand, it focuses on controlling the virtual environment, which has an impact on the physical environment.

All concepts from the general IoT architecture, as depicted in Fig. 1, which are not used in the specialized metamodel in Fig. 3, were excluded as the specialization of this model, though not garden specific, was driven by the specific requirements of the application at hand. This implies that the specialized metamodel was tailored to meet the unique requirements and constraints of the current application (Smart Garden, Sect. 3.2) domain, thereby optimizing the model for its intended use. The focus on the specific application allowed for a more precise and efficient representation of the necessary components, ensuring that the model could effectively support the implementation and operation of the IoT system within the defined context.

Furthermore, the flexibility of the IoT Open-Source Platform ensures that any missing concepts or functionalities can be integrated as needed, either through direct inclusion or by extending the current modeling method. This adaptability is crucial for addressing the evolving needs of IoT applications, allowing the system to remain relevant and effective as new requirements and technologies emerge.

The *Infrastructure and Software* part is hosted by OMiPOB, which stands for OMiLAB Physical Objects, and is part of the Open Models initiative Laboratory

(OMiLAB) [33]. It provides the framework for the experiment—the setup of a scenario-based, conceptual, and physical infrastructure in which various activities can be executed [32].

3.2 A DSL for Smart Gardens

In this section, we introduce the domain specific language used for the smart garden example (and initial validation) of the HESTIA framework.

The model includes graphical notations that represent various elements crucial to the system's operation. These elements encompass individuals, facilities, as well as the sensors and actuators that interact with the environment. Each notation is designed to visually communicate the role and function of these components within the system, ensuring clarity and facilitating user interaction (Table 1).

To ensure that the garden behaves as desired by the user, in addition to the IoT platform, a control system is necessary. This system operates based on individual rules, each consisting of at least one condition that must be checked. Depending

Table 1 Visual representations of the concepts and relation concept classes

Concept	Visualisation	Concept	Visualisation
Adult		Camera	
Car		Cat	
Child		Dog	
External		Garage	
Greenhouse		Home	
Humidity		Irrigation_System	
Lawn		Lawn_Mower	
Light		Motion	
Outdoor_Lighting		Plant	
Playground		Pool	
Rain		RFID_Reader	
RFID_Tag		Security_Lock	
Street		Sun_Protection	
Temperature		Tree	
belongs to			

Table 2 Visual representations of the concept classes

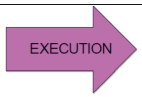
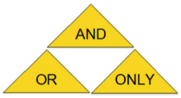
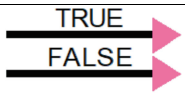
Concept	Visualisation	Description
Execution		All rules that are linked to the fire relationship are executed when the Execution button is triggered. The existing rules that are not connected are ignored.
Rule		The rule is connected to an expression. This is done with the relationship connects.
Expression		A distinction can be made between ONLY, AND and OR in the expression. ONLY means that only one comparison can be checked. With AND, all comparisons must return TRUE. With OR, at least one of the comparisons must return TRUE.
Comparison		Each expression has at least one comparison. Only one comparison may be connected to the ONLY expression.
Action		Contains the action to be executed.

Table 3 Visual representations of the relation concept classes

Concept	Visualisation	Description
fire		Constraint: Connection only possible with Execution → Rule
connects		Constraint: Connection only possible with Rule → Expression
consist of		Constraint: Connection only possible with Expression → Comparison
execute		Standard type is TRUE. Constraint: Connection only possible with Expression → Action
next Action		Refers to the next action that must be performed. Constraint: Connection only possible with Action → Action

on the evaluation of these conditions, specific actions are executed according to the logic of the system (**if-then-else**). The elements required for this control system are depicted graphically in Table 2 and are relations in Table 3.

Short-circuit evaluation is an optimization technique used in evaluating logical expressions to improve efficiency. When dealing with OR expressions, the evaluation can stop as soon as one comparison returns TRUE because only one TRUE result is necessary to make the entire expression TRUE. This prevents the need to evaluate any remaining comparisons. For instance, if a condition in an OR expression returns TRUE early on, the subsequent comparisons are skipped, saving computational resources. Conversely, for AND expressions, all conditions must be TRUE for the entire expression to be TRUE. Therefore, if any single condition returns FALSE, the evaluation can stop immediately, as the overall result cannot be TRUE. This immediate termination upon encountering a FALSE result avoids unnecessary evaluations of the remaining conditions. In some cases, a condition

within an expression might return TRUE or FALSE without leading to a direct action based on that single result. The decision to take action depends on the overall logical structure and the specific requirements of the evaluation process.

4 Proof of Concept: An Example Garden

Instead of a real garden, we used a miniature garden model so that we could control and vary the conditions without affecting a real-world garden. The used miniature garden model is located on the premises of OMiLAB Vienna.

A model serves as an illustration of the real world [34, p. 129–133]. The following elements are used to reproduce the reality of a garden in this miniature model. Figures such as adults and children are used as representative human actors. Animal actors are represented by a dog and a cat. The model also includes a house, a greenhouse, a garage and the corresponding vehicle, a pool, and a playground. The garden is being completed with more trees, a road, and a lawn.

From the technical point of view also shown in Fig. 4, we used technical components such as plug-in boards, plug-in cables, resistors, and LEDs, along with sensors like the DHT11 for temperature and humidity, the Mifare RC522 RFID Kit for access management, and controllers including the WEMOS TTGO ESP32 and Arduino Uno R3 with Ethernet Shield W5100 V1. Communication between

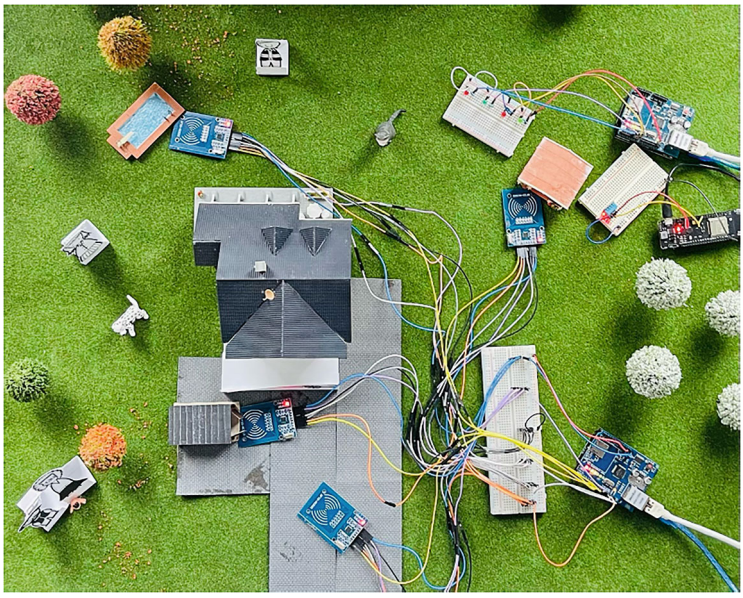


Fig. 4 Smart garden with technical components

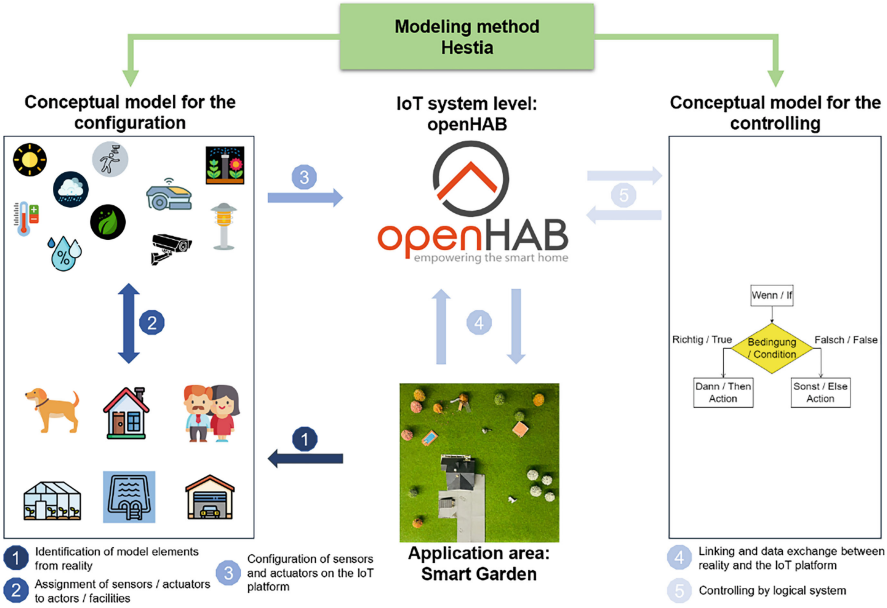


Fig. 5 Divisions of the HESTIA garden modeling language

the controllers and the openHAB smart home platform takes place via the MQTT Mosquitto Broker. This broker is located on the Raspberry Pi and is based on the publish/subscribe communication model.

Once all the physical components have been assembled and connected to openHAB, the final state is as shown in Fig. 4. In this setup, the smart garden utilizes sensors to monitor environmental conditions and actuators to control access and other functions. All physical elements are virtually represented in the openHAB platform, allowing for comprehensive control of the garden's behavior directly through the platform. For instance, users can easily switch the lighting on and off.

As mentioned above, a miniature model of a smart garden (see Fig. 4) is used as a base to provide a reference to reality. The example garden is based on the HESTIA framework and follows the structure as shown in Fig. 5.

The sensors and actuators are installed at the physical level in the real world. Based on this, all relevant elements from the area of application are identified and abstracted in the *conceptual model for the configuration*. In a further step, the sensors and actuators are assigned to actors and facilities. The necessary model elements also provide the basis for its representation and configuration on the IoT platform.

Due to the free availability of the platform and the free access to the source code and in order to ensure reproducibility, the open-source platform openHAB was specifically chosen as a base. The IoT platform interacts directly with the real world, the smart garden. Data is exchanged between them. Control of the

IoT platform is triggered by the rules defined in the *conceptual model for the controlling*. The sensors and actuators associated with the platform, in turn, have an effect on the real world. The two conceptual models together form the HESTIA garden modeling language (i.e., a DSL for gardens). This has been implemented on the ADOxx metamodeling platform, which is provided by OMiLAB. The ADOxx toolkit supports views that integrate concepts from various models, like application domain and rule models, through the corresponding model types. Interrelationships between models are supported by ADOxx.

Together with the architecture, the platform, and the conceptual modeling method, this comprises the HESTIA framework.

The scenario layer, depicted at the top level in Fig. 2, represents the real-world context in which the experiment is conducted. In this case, the application domain is the smart garden, where scenarios encompass both the actors involved and the realistic actions they may perform, enabling precise simulation and control of the garden environment. Two possible scenarios, Sects. 4.1 and 4.2, are presented subsequently.

4.1 Scenario I

The primary goal of this scenario is to ensure pool safety by managing the pool cover based on the presence of children and animals, as well as environmental conditions such as rain. This goal can be expressed as a user's need for the garden to behave as follows: *For safety reasons, the pool safety cover should be closed whenever children or animals are present in the garden. However, if the garden is empty during the day, it is acceptable to leave the cover open. Additionally, the cover should always be closed if it starts to rain, ensuring safety and preventing water accumulation.*

In the conceptual modeling layer, the relevant model elements are interconnected within the application domain model type (see Fig. 6). The pool is equipped with several key components, including a safety cover to ensure the pool is covered when necessary, an RFID reader and tags for access control to detect the presence of children and animals, and environmental sensors such as a rain sensor and a light sensor to monitor weather conditions.

Each of these elements is linked with the IoT platform through a unique identifier (*IdentificationName*) available in the openHAB system, allowing the system to recognize and control each device accurately. This identifier is customizable, enabling users to define it themselves on the IoT platform according to their preferences and needs.

The control/logical system for the pool safety cover is defined by a set of rules using an if-then-else structure that determines its operation based on specific conditions.

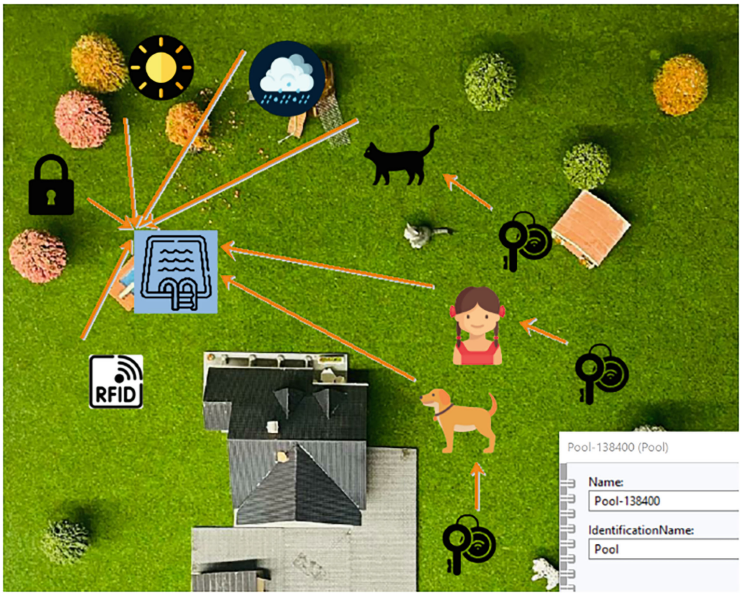


Fig. 6 Conceptual model for configuration

Rule 1: Presence Detection
If a child, dog, or cat is detected near the pool, the pool safety cover will close. If none of these are present, the cover will remain open.

```
1 if Comparison-1: The child is present.
2     or Comparison-2: The dog is present.
      or Comparison-3: The cat is present.
3 then
4     Action-1: The pool safety cover is being closed.
5 else
6     Action-2: The pool safety cover remains open.
```

Rule 2: Light Intensity and Absence Detection
If the light intensity is above 100 lux and no children, dogs, or cats are present, the pool safety cover will open. If these conditions are not met, the cover will remain closed.

```
1 if Comparison-4: The light intensity is over 100 lux.
2     and Comparison-5: The child is absent.
      and Comparison-6: The dog is absent.
      and Comparison-7: The cat is absent.
3 then
4     Action-3: The pool safety cover will be opened.
5 else
6     Action-4: The pool safety cover remains closed.
```

Rule 3: Rain Detection
If it is raining, the pool safety cover will close.

```
1 if Comparison-8: It's raining.
  then
2     Action-5: The pool safety cover is being closed.
```

The individual rules are created using the notation of Tables 2 and 3 in the model type Rules (see Fig. 8). In Fig. 7, the Comparison-8 and the Action-5 are defined from Rule 3.

In the physical layer, the initial state recorded by openHAB includes the presence of a child, the absence of a dog, rainy conditions, light intensity at 1500 lux, and the pool safety cover being open.

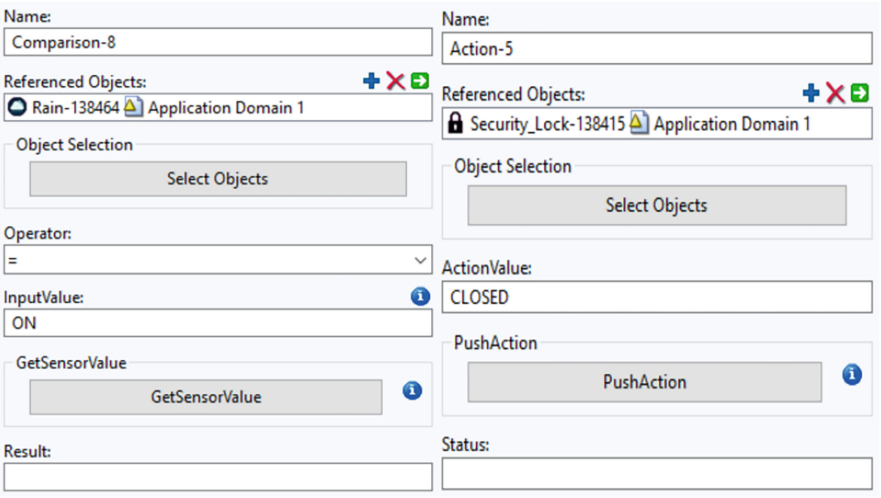


Fig. 7 Comparison and Action: Rule 3

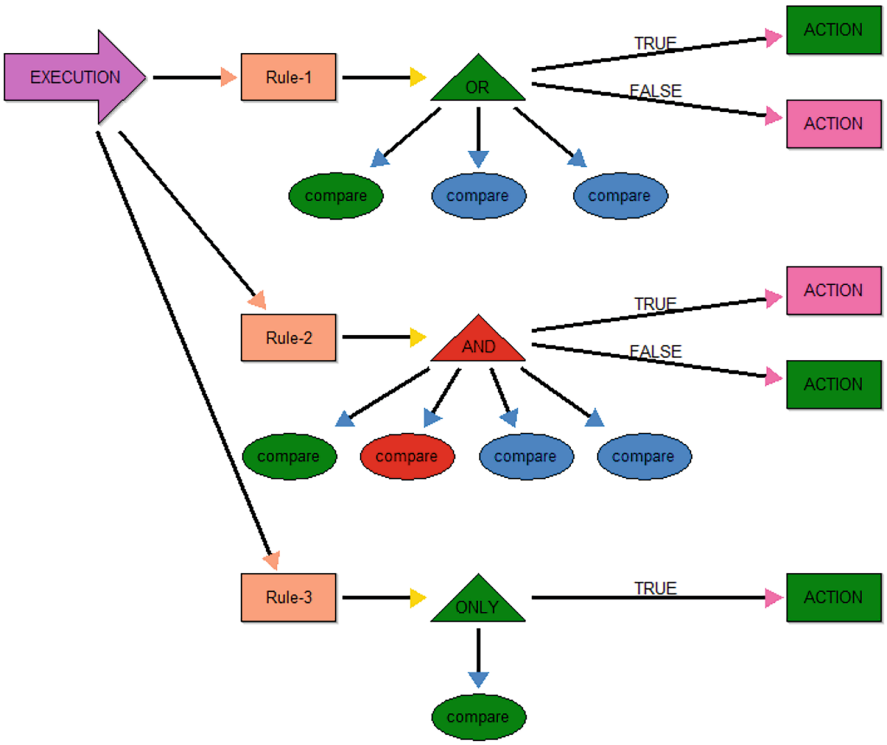


Fig. 8 Conceptual model for controlling: after execution

The rules are executed based on the current status of these elements. To increase the clarity of how the concepts Comparison and Action were executed, they are visually differentiated by a color-coding system: green is used to indicate a successful result if the value is TRUE, and red indicates an unsuccessful result if the value is FALSE. This visual distinction enables an intuitive understanding of the results of the rule execution, which can be seen in Fig. 8:

Rule 1 executes successfully as a child is detected, leading to the closure of the pool safety cover.

Rule 2 does not fully execute because the condition of the child's absence is not met.

Rule 3 executes successfully due to the detection of rain, confirming the action to close the pool safety cover.

The final state of the pool safety cover after the execution of these rules is illustrated in Fig. 9 as displayed in openHAB.

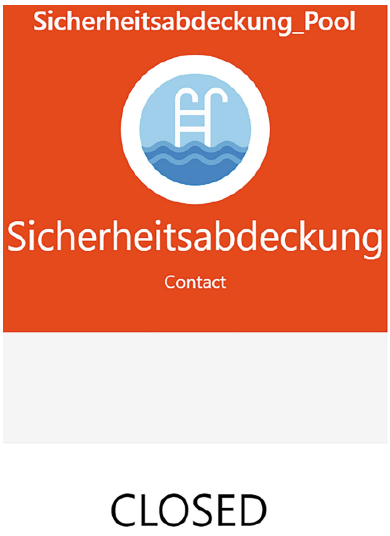


Fig. 9 openHAB final state

4.2 Scenario II

The objective of this scenario is to enhance the security of the garage and automate the opening of the garage door based on specific conditions, such as motion detection, light intensity, and the presence of an authorized vehicle. This goal can be expressed as a user's need for the garden to behave as follows: *Since the garage is some distance from the house, it's not possible to see if someone is in there without authorization. To address this, it would be helpful to illuminate the dark corners of the garage at night. Another option would be to check what has happened at a later time. Additionally, it would be advantageous if the garage door automatically opens when you drive onto the forecourt with your vehicle.*

In the conceptual modeling layer, the relevant model elements are interconnected within the Application Domain model type.

The garage is equipped with several key components (see Fig. 10):

Security camera: monitors the garage for unauthorized access

Light sensor: detects the ambient light intensity

Motion sensor: identifies movement around the garage

RFID reader and tags: detects the presence of an authorized vehicle

Garage door: automates the opening and closing based on certain conditions

Each of these components is linked with the IoT platform through a unique identifier (*IdentificationName*) available in the openHAB system, allowing the system to recognize and control each device accurately.

The control/logical system for the garage is defined by a set of rules using an if-then-else structure that determines its operation based on specific conditions.

Fig. 10 Application domain

Rule 1: Motion and Light Detection
If motion is detected and the light intensity is less than or equal to 60 lux, the security camera and all garage lights will turn on. If no motion is detected, the security camera and lights will remain off.

```
1 if Comparison-1: Motion was recognized.
2     and Comparison-2: The light intensity is less than or equal to 60 lux.
3 then
4     Action-1: Security camera is turned on. Action-2: Lamp_1 is switched
        on. Action-3: Lamp_2 is switched on. Action-4: Lamp_3 is switched
        on. Action-5: Lamp_4 is switched on.
```

Rule 2: Vehicle Presence Detection
If an authorized vehicle is detected on the forecourt, the garage door will open. If the vehicle is not detected, the garage door will remain closed.

```
1 if Comparison-3: The vehicle is located on the forecourt.
    then
2     Action-6: Garage door opens.
3 else
4     Action-7: Garage door remains closed.
```

The individual rules are created using the notation of Tables 2 and 3 in the model type Rules (see Fig. 8). In Fig. 11, the Comparison-3 and both Action-6 and 7 are defined from Rule 2.

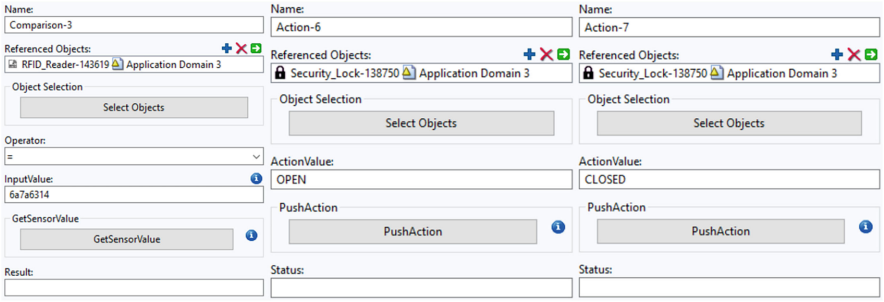


Fig. 11 Comparison and Actions: Rule 2

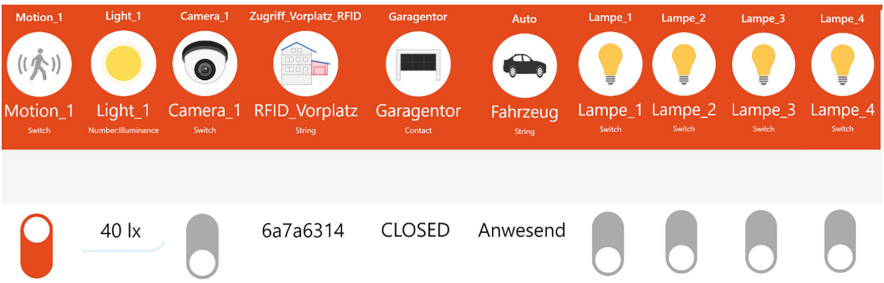


Fig. 12 Initial situation

In the physical layer, the initial state recorded by openHAB includes the presence of motion, light intensity at 40 lux, and the garage door being closed. Further conditions can be seen in Fig. 12.

The rules are executed based on the current status of these elements, and the results are as follows (see Fig. 13). The color-coding system is also evident here, providing a simplified understanding that all comparisons and actions were successfully executed, returning a value of TRUE.

After executing the rules, the security camera and lights are turned on, and the garage door is open. These environmental conditions can also be found in openHAB (see Fig. 14).

As a result of executing these rules, the actuators in the miniature model are controlled according to the physical layer, leading to all four lamps being switched on, as shown in Fig. 15.

5 Reflection

The proof of concept (PoC) in the HESTIA framework discloses the feasibility and potential of garden automation within the broader context of the Internet of Things

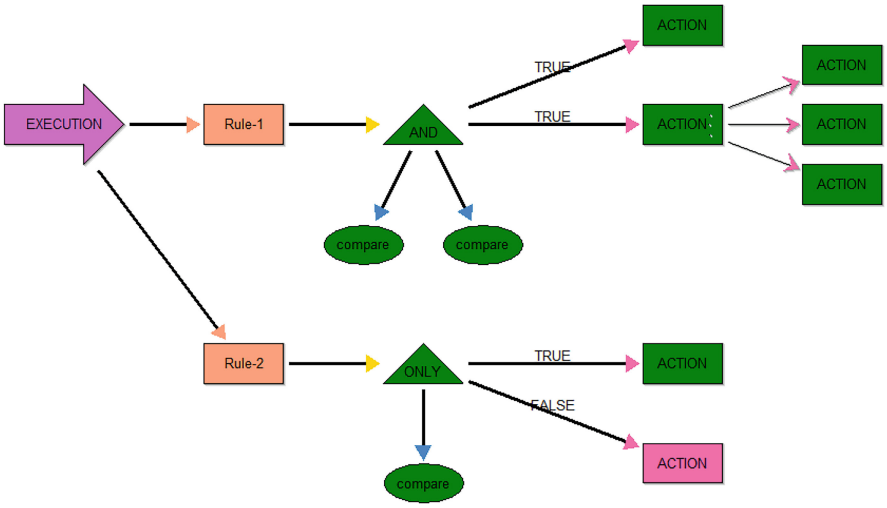


Fig. 13 Conceptual model for controlling: after execution

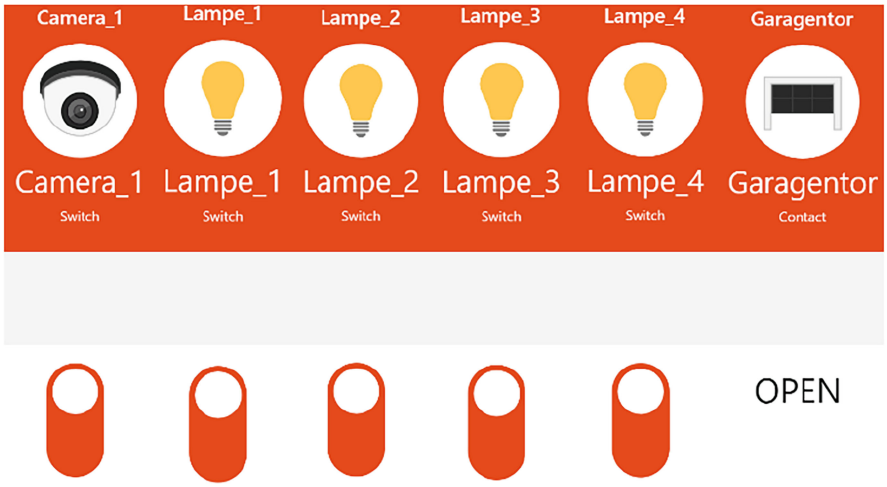


Fig. 14 openHAB end state

(IoT). This PoC focuses on the small-scale implementation of a smart garden, which acts as an exemplary model representation for large-scale applications. By leveraging the HESTIA framework, which includes an overall architecture for IoT, a domain-specific conceptual modeling with a user-friendly domain-specific language, and the associated ADOxx platform, the proof of concept effectively bridges the gap between technical complexities and user requirements.

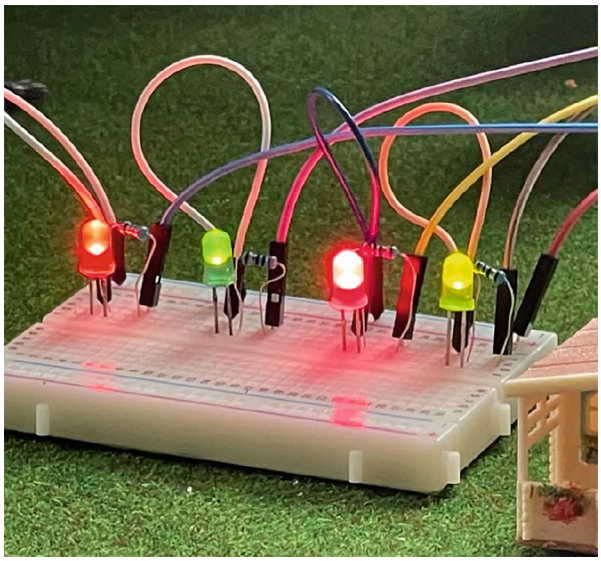


Fig. 15 Smart garden end state

Feasibility and Functionality: The PoC exposed how HESTIA can abstract technicalities to allow user interaction with IoT devices in a more intuitive and meaningful experience. The model of the miniature garden as depicted in Fig. 4 was used for the PoC to allow experiments under control and the testing of capabilities without affecting real-world environments.

User-Friendly Control: The domain-specific language developed for smart gardens proved to be an effective tool for users to define control and behavioral rules. This way, it will decrease the burden of interaction with the IoT platform even for those who are not so technically savvy. The implementation of rules for garden automation, such as managing pool safety covers and garage security, demonstrated the practical applications of the framework.

Scalability and Generalization: Although the PoC was conducted on a small scale, the findings suggest that the same principles can be applied to larger-scale IoT implementations. The modular and adaptable nature of the HESTIA framework supports scalability. The conceptual models and methodologies used in the PoC can be extended to other domains within smart homes and cities, showcasing the versatility of the framework.

Future research efforts should also incorporate advanced automation techniques, and artificial intelligence into the HESTIA framework could pave the way for more sophisticated and autonomous decision-making processes. By exploring the potential of machine learning algorithms, future studies could optimize garden management and other smart home functions, tailoring them to user behavior and environmental conditions for greater efficiency and personalization.

In addition, further research into improving the user interface and overall user experience will be crucial to ensure that the HESTIA framework remains accessible to a wider audience, meaning to include more actors, facilities, sensors, and actuators. Users need change and expand, as well as the range of sensors and actuators on market. This will make the domain-specific language even more intuitive and shorten the learning curve for new users. The framework can thus improve its acceptance and user-friendliness and ultimately make advanced IoT technology more accessible to all users.

What is also important to realize is that the creation of a model of, for instance, a house or a garden for the purpose of, e.g., home automation, is not an easy task, even when using modeling symbols that are close to a user's intuition. As also acknowledged in, e.g., [35], creating such models requires a new skill profile. As such, a major challenge is to enable, and provide automated support, for non-modeling-experts to engage in the required modeling efforts. This challenge has fueled earlier efforts to make modeling strategies more explicit [36–38], as well as experiments with the concept of *natural modeling* [39, 40], which has also been echoed by the more recent notion of *grassroots modeling* [41].

6 Conclusion

HESTIA has effectively shown automation in an IoT ecosystem smart garden through the proof-of-concept implementation. Abstracting complex technical details and making them user-friendly, HESTIA closes the gap between intricate IoT configurations and practical, everyday applications. In the research, the miniature model of a garden showed the feasibility and functionality of the framework. It shows that even small implementations could serve as a relevant prototypes toward larger applications.

A significant achievement of the HESTIA framework lies in its domain-specific language, which enables users to define control and behavioral rules intuitively. This approach not only simplifies the interaction with IoT devices but also makes advanced technology accessible to non-technical users. The successful execution of various automation scenarios, such as managing pool safety and enhancing garage security, highlights the practical benefits and versatility of the framework. This approach not only simplifies the interaction with IoT devices but also makes advanced technology accessible to non-technical users. The successful execution of various automation scenarios, such as managing pool safety and enhancing garage security, highlights the practical benefits and versatility of the framework.

Although this proof of concept has been conducted only with respect to a small-scale garden model, the underlying principles and methodologies are generalizable to more extensive smart home and city environments. This scalability is crucial for the broader adoption of IoT technologies, making HESTIA a valuable framework for future developments in this field.

In conclusion, the HESTIA framework represents a significant step forward in making IoT technologies more accessible and practical for everyday applications. Its successful proof of concept in garden automation serves as a promising foundation for future advancements, paving the way for more sophisticated and user-friendly smart environments.

Tool Download: https://www.omilab.org/hestia

References

1. United Nations—Department of Economic and Social Affairs: *68% of the World Population Projected to Live in Urban Areas by 2050, Says UN*. https://www.un.org/development/desa/en/news/population/2018-revision-of-world-urbanization-prospects.html [Accessed 23.05.2023] (2018)
2. Muck, C., Voelz, A., Amlashi, D.M., Karagiannis, D.: Citizens as developers and consumers of smart city services: a drone tour guide case. In: WebAndTheCity: 8th International Workshop on Web and Smart Cities. Companion Proceedings of the Web Conference 2022, pp. 1228–1236. Association for Computing Machinery, New York (2022). https://doi.org/10.1145/3487553.3524848
3. KPMG: *Immobilien + Innovationen: Themen, Trends und Technologien für die digitale Immobilienwirtschaft*. https://kpmg.com/de/de/home/branchen/real-estate/digital-real-estate/immobilien-innovationen/smart-infrastructure.html [Accessed 23.05.2023] (2021)
4. Gunge, V.S., Yalagi, P.S.: Smart home automation: A literature review. IJCA Proceedings on National Seminar on Recent Trends in Data Mining **RTDM 2016**(1), 6–10 (2016)
5. Ahmad, M.O., Ahad, M.A., Alam, M.A., Siddiqui, F., Casalino, G.: Cyber-physical systems and smart cities in India: Opportunities, issues, and challenges. Sensors (Basel, Switzerland) **21**(22), 7714 (2021). https://doi.org/10.3390/s21227714
6. Deutsche Forschungsgemeinschaft: *DFG fördert elf neue Sonderforschungsbereiche*. https://www.dfg.de/service/presse/pressemitteilungen/2023/pressemitteilung_nr_14/index.html [Accessed 23.05.2023] (2023)
7. Greer, C., Burns, M., Wollman, D., Griffor, E.: Cyber-Physical Systems and Internet of Things. In: Special Publication (NIST SP), National Institute of Standards and Technology, Gaithersburg, MD. https://doi.org/10.6028/NIST.SP.1900-202 (2019)
8. Grosu, R.: *Cyber-Physical Systems: Die Ganze Welt Wird smart*. https://ti.tuwien.ac.at/cps/people/grosu/tuw-cps-week-de.pdf [Accessed 23.05.2023] (2016)
9. Plessis, J.C.: Smart World Cities in the 21st Century. Agnes Mainka. Boston: Walter de Gruyter, 2018. 266 pp. (ISBN 978–3–11–057525–5). J. Assoc. Inf. Sci. Technol. **71**(2), 243–244 (2020). https://doi.org/10.1002/asi.24225
10. Wegner, P.: Iot platform companies landscape 2021/2022: Market consolidation has started. IoT Analytics GmbH (2021). https://iot-analytics.com/iot-platform-companies-landscape/[Accessed 26.03.2023]
11. Becks, E., Zdankin, P., Matkovic, V., Weis, T.: Complexity of smart home setups: A qualitative user study on smart home assistance and implications on technical requirements. Technologies **11**(1) (2023). https://doi.org/10.3390/technologies11010009
12. Guarino, N., Guizzardi, G., Mylopoulos, J.: On the philosophical foundations of conceptual models. Information Modelling and Knowledge Bases XXXI **321**, 1–15 (2020). https://doi.org/10.3233/FAIA200002

13. Proper, H.A., Guizzardi, G.: On domain conceptualization. In: Aveiro, D., Guizzardi, G., Pergl, R., Proper, H.A. (eds.) Advances in Enterprise Engineering XIV—10th Enterprise Engineering Working Conference, EEWC 2020, Bozen-Bolzano, Italy, September 28, October 19, and November 9–10, 2020, Revised Selected Papers. Lecture Notes in Business Information Processing, vol. 411, pp. 49–69. Springer, Berlin (2021). https://doi.org/10.1007/978-3-030-74196-9_4
14. Lukyanenko, R., Storey, V.C., Pastor, O.: System: A core conceptual modeling construct for capturing complexity. Data Knowl. Eng. **141**, 102062 (2022). https://doi.org/10.1016/j.datak.2022.102062
15. Fowler, M.G.: Analysis Patterns—Reusable Object Models. Addison-Wesley Longman Publishing Co., Inc. eBooks, USA (1996)
16. OMG: Unified Modeling Language (UML), Version 2.5. OMG (2015). http://www.omg.org/spec/UML/2.5/
17. Tekinerdogan, B., Köksal, Ö., Çelik, T.: Data Distribution Service-Based Architecture Design for the Internet of Things Systems. In: Mahmood, Z. (ed.) Connected Environments for the Internet of Things: Challenges and Solutions, pp. 269–285. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70102-8_13
18. Paolone, G., Iachetti, D., Paesani, R., Pilotti, F., Marinelli, M., Di Felice, P.: A holistic overview of the Internet of Things ecosystem. IoT **3**(4), 398–434 (2022). https://doi.org/10.3390/iot3040022
19. Haller, S., Serbanati, A., Bauer, M., Carrez, F.: A domain model for the Internet of Things. In: 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing, pp. 411–417 (2013). IEEE, New York. https://doi.org/10.1109/GreenCom-iThings-CPSCom.2013.87
20. Bassi, A., Bauer, M., Fiedler, M., Kramp, T., Van Kranenburg, R., Lange, S., Meissner, S.: Enabling things to talk: designing IoT solutions with the IoT architectural reference model. Springer, Berlin (2013). https://doi.org/10.1007/978-3-642-40403-0
21. Haller, S.: The things in the Internet of Things. In: Internet of Things Conference (IoT2010) 2010, Tokyo, 29 November–1 December 2010 (2010)
22. Serbanati, A., Medaglia, C.M., Ceipidor, U.B., Turcu, C.: Building blocks of the Internet of Things: state of the art and beyond. Deploying RFID-Challenges, Solutions, and Open Issues **395**, 116–124 (2011). https://doi.org/10.5772/19997
23. Tubaishat, M., Madria, S.: Sensor networks: an overview. IEEE Potentials **22**(2), 20–23 (2003). https://doi.org/10.1109/MP.2003.1197877
24. Patel, P., Cassou, D.: Enabling high-level application development for the Internet of Things. J. Syst. Softw. **103**, 62–84 (2015). https://doi.org/10.1016/jjss.2015.01.027
25. Barnaghi, P.: Sensei: an architecture for the real world internet. In: First International Workshop on Semantic Interoperability for Smart Spaces (SISS 2010) June 2010, Riccione, Italy (2010). https://iscc2010.ieee-iscc.org/web_pages/SISS/presentation/keynote.pdf [Accessed 18.04.2023]
26. Di Martino, B., Rak, M., Ficco, M., Esposito, A., Maisto, S.A., Nacchia, S.: Internet of things reference architectures, security and interoperability: a survey. Internet Things **1**, 99–112 (2018). https://doi.org/10.1016/j.iot.2018.08.008
27. Ciccozzi, F., Crnkovic, I., Di Ruscio, D., Malavolta, I., Pelliccione, P., Spalazzese, R.: Model-driven engineering for mission-critical IoT systems. IEEE Softw. **34**(1), 46–53 (2017). https://doi.org/10.1109/MS.2017.1
28. Corradini, F., Fedeli, A., Fornari, F., Polini, A., Re, B., Ruschioni, L.: X-IoT: a model-driven approach to support IoT application portability across IoT platforms. Computing 1–25 (2023). https://doi.org/10.1007/s00607-023-01155-z
29. Rehman, S.u., Ceglia, M., Siddiqui, S., Gruhn, V.: Towards an Importance of Security for Cyber-Physical Systems/Internet-of-Things. In: Proceedings of the 8th International Conference on Software and Information Engineering. ICSIE '19, pp. 151–155. Association for Computing Machinery, New York (2019). https://doi.org/10.1145/3328833.3328855

30. Ihirwe, F., Di Ruscio, D., Mazzini, S., Pierini, P., Pierantonio, A.: Low-code engineering for internet of things: a state of research. In: Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings, pp. 1–8 (2020). https://doi.org/10.1145/3417990.3420208
31. International Electrotechnical Commission: *ISO/IEC 30141:2018 Internet of Things (IoT)-Reference Architecture*. https://standards.iso.org/ittf/PubliclyAvailableStandards/c065695_ISO_IEC_30141_2018(E).zip [Accessed 20.04.2023] (2018)
32. Woitsch, R.: Industrial digital environments in action: the OMiLAB innovation corner. In: Grabis, J., Bork, D. (eds.) The Practice of Enterprise Modeling: 13th IFIP Working Conference, PoEM 2020, Riga, Latvia, November 25–27, 2020, Proceedings 13, pp. 8–22 (2020). Springer, Cham. https://doi.org/10.1007/978-3-030-63479-7_2
33. Karagiannis, D., Muck, C.: *OMiLAB Physical Objects (OMiPOB)*. https://www.omilab.org/assets/docs/OMiROB_description_draft.pdf [Accessed 24.05.2023] (2017)
34. Stachowiak, H.: Allgemeine Modelltheorie. Springer, Wien (1973)
35. Karagiannis, D., Buchmann, R.A., Boucher, X., Cavalieri, S., Florea, A., Kiritsis, D., Lee, M.: OMiLAB: A smart innovation environment for digital engineers. In: Camarinha-Matos, L.M., Afsarmanesh, H., Bas, Á.O. (eds.) Boosting Collaborative Networks 4.0—21st IFIP WG 5.5 Working Conference on Virtual Enterprises, PRO-VE 2020, Valencia, Spain, November 23–25, 2020, Proceedings. IFIP Advances in Information and Communication Technology, vol. 598, pp. 273–282. Springer, Berlin (2020). https://doi.org/10.1007/978-3-030-62412-5_23
36. Hoppenbrouwers, S.J.B.A., Lindeman, L., Proper, H.A.: Capturing modeling processes—towards the MoDial modeling laboratory. In: Meersman, R., Tari, Z., Herrero, P. (eds.) On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops, OTM Confederated International Workshops and Posters, AWeSOMe, CAMS, COMINF, IS, KSinBIT, MIOS-CIAO, MONET, OnToContent, ORM, PerSys, OTM Academy Doctoral Consortium, RDDS, SWWS, and SeBGIS 2006, Montpellier, France, October 29–November 3, 2006. Proceedings, Part II. Lecture Notes in Computer Science, vol. 4278, pp. 1242–1252. Springer, Berlin (2006). https://doi.org/10.1007/11915072_27
37. Hoppenbrouwers, S.J.B.A., Proper, H.A., Weide, T.P.: A fundamental view on the process of conceptual modeling. In: Delcambre, L.M.L., Kop, C., Mayr, H.C., Mylopoulos, J., Pastor López, Ó. (eds.) Conceptual Modeling—ER 2005, 24th International Conference on Conceptual Modeling, Klagenfurt, Austria, October 24-28, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3716, pp. 128–143. Springer, Berlin (2005). https://doi.org/10.1007/11568322_9
38. Hoppenbrouwers, S.J.B.A., Proper, H.A., Weide, T.P.: Towards explicit strategies for modeling. In: Halpin, T.A., Siau, K., Krogstie, J. (eds.) Proceedings of the 10th International Workshop on Exploring Modeling Methods for Systems Analysis and Design, EMMSAD 2005, Held in Conjunction with the 17th Conference on Advanced Information Systems (CAiSE 2005), Porto, Portugal, 13-14 June, 2005. CEUR Workshop Proceedings, vol. 363, pp. 99–106. CEUR-WS.org, Aachen, Germany (2008). https://ceur-ws.org/Vol-363/paper9.pdf
39. Bjeković, M., Sottet, J.-S., Favre, J.-M., Proper, H.A.: A framework for natural enterprise modelling. In: IEEE 15th Conference on Business Informatics, CBI 2013, Vienna, Austria, July 15–18, 2013, pp. 79–84. IEEE Computer Society, Washington DC, USA (2013). https://doi.org/10.1109/CBI.2013.20
40. Zarwin, Z., Bjeković, M., Favre, J.-M., Sottet, J.-S., Proper, H.A.: Natural modelling. J. Object Technol. **13**(3), 4–136 (2014). https://doi.org/10.5381/JOT.2014.13.3.A4
41. Sandkuhl, K., Fill, H.-G., Hoppenbrouwers, S.J.B.A., Krogstie, J., Matthes, F., Opdahl, A.L., Schwabe, G., Uludağ, Ö., Winter, R.: From expert discipline to common practice: a vision and research agenda for extending the reach of enterprise modeling. Bus. Inf. Syst. Eng. **60**(1), 69–80 (2018). https://doi.org/10.1007/s12599-017-0516-y