Computer Science and Information Systems 20(4):1639–1659 https://doi.org/10.2298/CSIS230328049L

A Faceted Discovery Model Architecture for Cyber-Physical Systems in the Web of Things

Juan Alberto Llopis, Manel Mena, Javier Criado, Luis Iribarne, and Antonio Corral

Applied Computing Group (TIC-211), *University of Almería* Almería, Spain
{jalbertollopis, manel.mena, javi.criado, luis.iribarne, acorral}@ual.es

Abstract. A Cyber-Physical System is a set of heterogeneous devices that integrates computational and digital capabilities with their physical system. As technology evolves to facilitate human tasks, more complex Cyber-Physical Systems are being developed, even integrating them with web technologies (Web of Things), e.g., in the context of the Web of Things, supporting smart city scenarios with thousands of devices available to be discovered online. In these complex solutions, some capabilities related to locating, registering, and consulting devices must be provided to adapt to the continuous changes in Cyber-Physical Systems. Suitable capabilities could be using natural language queries, automatically describing and discovering new devices, or locating devices in different subsystems. This paper[1] proposes a discovery model architecture for Cyber-Physical Systems based on the Web of Things, including proactive discovery, recommendation, federation, and query expansion. In an example scenario, the proposed architecture has been implemented with different topologies using Edge Computing facilities to interact and manage Cyber-Physical Systems. The results show that the capabilities of the discovery model architecture facilitate the discovery of CPSs in different smart environments.

Keywords: Discovery Model, Web of Things, Cyber-Physical Systems, WoT Lab.

1. Introduction

Cyber-Physical Systems (CPSs) are characterized by integrating physical devices with computational and digital capabilities, working as a bridge between the physical and digital world [36, 39]. As the Internet of Things (IoT) is focused on communication between devices, CPSs include the interconnection and collaboration of IoT devices for improving the communication between CPSs [17, 30]. Therefore, CPSs consist of heterogeneous devices that make up an ecosystem to solve problems in smart environments, e.g., appliances, smart watches, or sensors working with different technologies and protocols. The Web of Things (WoT) initiative [23] attempts to bring all those different devices of the CPSs together in a reference framework supported by the World Wide Web Consortium (W3C). The WoT aims to establish an abstraction layer based on web technologies for managing Internet of Things (IoT) devices, thus approaching the interconnection problem between heterogeneous devices in smart environments.

Smart environments are scenarios supported by CPSs, such as Smart Cities, Smart Homes, or Smart Healthcare, equipped with computing power and coordinated by intelligent systems to give the environment intelligence for supporting and reducing human

[1] This is an extended version of The 14th International Conference on Management of Digital EcoSystems conference paper "Towards a Discovery Model for the Web of Things".

interaction [22]. Smart environments are evolving into more challenging ecosystems that use a higher number of devices with complex functionality. For instance, a smart garbage collection in a Smart City, where trucks, containers, traffic, and traffic lights are monitored to find time-optimal dynamic routes [4]. In this sense, discovering and managing devices of this kind of smart scenarios and CPSs can be difficult.

A CPS ecosystem may integrate devices from different locations or subsystems, e.g., devices in different buildings; it may require the processing of complex or abstract queries, e.g., natural language queries in a Smart Home or commercial building; or it may need to discover automatically devices, e.g., a laboratory changing continuously. Current Discovery Services in smart scenarios focus on managing many CPSs, but in terms of query efficiency. Devices register themselves into the Discovery Service, and users search for them using syntactic and semantic queries. However, with the evolution of CPSs and smart environments, discovery should include more capabilities to follow the continuous changes of the CPSs. These capabilities must adapt to the requirements when managing and searching for CPSs, and the discovery should be prepared to adopt new faceted capabilities progressively for future needs. A faceted perspective of the proposal facilitates further flexible, scalable, and open features of the architecture of the discovery model.

By capabilities of a Discovery Service, we mean features that are complex enough to be developed alone and support the main task of discovering CPSs. Normally, these features are not included in the Discovery Service system, the Discovery Service is developed as simply as possible in terms of functionality, and the capabilities are developed as individual systems unrelated to the Discovery Service. Nevertheless, with the evolution of CPSs, Discovery Services need additional capabilities as part of the main system to support the search and discovery. For instance, integrating Artificial Intelligence (AI) and federation capabilities with the Discovery Service to process natural language queries and search for devices in large-scale scenarios, where users interact with the system by voice commands. The Discovery Service must return devices located in different subsystems.

In this paper, we present a discovery model architecture for CPSs based on the Web of Things (WoT) that includes different capabilities in a faceted way: (1) discover proactively in a pull configuration, (2) recommending devices and services such as other Discovery Services using AI, (3) federated searches through a federation of Discovery Services, and (4) query expansion. Furthermore, as the discovery model architecture is based on the WoT, it can be used with other implementations and environments as long as the services are described following the Thing Description (TD) structure. The discovery model architecture with two of four capabilities (i.e., proactive discovery and recommendation) has been implemented in a real example scenario of a smart room with different topologies using Edge Computing facilities. The following research questions are addressed to identify the objectives as well as to approach the aforementioned facts:

- ***RQ1:*** *Would it be feasible to extend traditional Discovery Services for discovering cyber-physical devices?*
- ***RQ2:*** *Does implementing additional capabilities improve the discovery of CPSs?*

This paper extends one from *The 14th International Conference on Management of Digital EcoSystems* [27]. In [27], we proposed a discovery model for the WoT. In this paper, we extend that contribution with an architecture supported by four capabilities that complement the discovery model: recommendation system, proactive discovery, query

expansion, and federation (RQ2). Furthermore, we implement and validate the model in a smart room scenario using different topologies to experiment with the architecture in a real scenario with limitations, such as public access to CPS's information in a local network and communication between subsystems through secure connections (RQ1).

This article is structured as follows. Section 2 offers an overview of related projects about Discovery Services in CPSs and WoT, and describes the background information and the fundamentals required to understand the proposed discovery model architecture. Section 3 describes the architecture and the four capabilities complementing the discovery model. Section 4 shows the experimental scenario of a smart room to validate the proposed discovery model, while Section 5 discusses some advantages and limitations of the proposal. Finally, Section 6 presents the conclusions and outlines future work lines.

2. Literature Review

This section offers an overview of existing approaches related to the discovery model. Furthermore, a background is summarized to describe some of the main terms related to our approach.

2.1. Related Work

Related to discovery, there are different topics about discovering systems, e.g., discovery service, process discovery, or component discovery, among others [11]. In all of these topics exists techniques to discover the most relevant information. For instance, in [37], the authors apply process discovery algorithms on some of the event logs instead of applying them on the whole logs, thus increasing the performance in the discovery process. For component discovery, in [21], the authors propose a discovery, called trader, based on the Open Distributed Processing (ODP) trading model, for discovering and integrating COTS (Commercial Off-The-Shelf) components through a federation approach. Other techniques applied are the federation of directories to increase the range of solutions returned to the user [16] or the implementation of security mechanisms when searching for information [5]. In this paper, we focus on discovering CPSs by proposing the addition of capabilities to the Discovery Service to support the search process and to adapt the Discovery Service to the evolution of CPSs.

The proposed discovery model architecture is based on the W3C Discovery Service, which started with DiscoWoT, a Discovery Service for smart things [31]. Additionally, IoT Discovery Services are also relevant to this research work. One of the relevant contributions to discovering IoT devices is IoTCrawler [20]. In IoTCrawler, devices are discovered using the Context Information Management API (NGSI-LD) standard. Using the NGSI-LD standard, users can subscribe directly to the device data. Furthermore, NGSI-LD brokers can be interconnected, creating a federation where every broker can access the information of the other brokers. A feature of IoTCrawler similar to our approach is the layer structure used for the discovery. The discovery is deployed independently from the search, thus speeding up the search and discovery process. In our proposal, each capability is an independent system that supports the discovery system. This approach follows a modular structure, where more systems can be integrated; Furthermore, as they are independent systems, the performance and maintenance benefit from decoupling.

A more industrial approach for discovering IoT devices is the proposal of GS1 [12]. GS1 is an international organization that improves supply chain efficiency by creating standards such as the barcode and the EPC (Electronic Product Code) Tag Data. GS1 developed a Discovery Service for discovering RFID (Radio-frequency identification) items. The Discovery Service performs a broadcast to identify all RFID items deployed in a subnet. Furthermore, GS1 is developing a Discovery Service for choreographing repositories from different businesses that work in the same chain but are unrelated. For instance, connecting the information of a business that produces tomatoes with that of a business that uses tomatoes to create sauces. The solution of GS1 for discovering RFID items is similar to the proactive discovery capability. In both approaches, the Discovery Service scans the subnetwork to locate the items deployed. However, in our proposal, the discovery is focused on CPSs, where multiple communication protocols are involved.

Another approach is QoDisco [15], a semantic-based discovery service that stores the information of the devices following an ontology-based information model. The ontology extends the Semantic Sensor Network, the Semantic Actuator Network, part of the Standard Ontology for Ubiquitous and Pervasive Applications, and the Web Ontology Language for Services. QoDisco focuses on discovering IoT devices registered in repositories. The user selects the query from the repositories more relevant to the query, and QoDisco returns the address and topic of the broker with the desired information. QoDisco focuses on how to store and retrieve the information of IoT devices. Our approach focuses on facilitating the discovery of CPSs, thus proposing capabilities that improve the discovery in different scenarios. Nevertheless, the ideas of QoDisco about using the domain and searching for devices in the repositories could help us research the federation capability.

Regarding the discovery of WoT devices, WoTStore [38] is a framework based on the recommendation of the W3C for managing and deploying *Things* and applications related to the *Things*. WoTStore is microservice-oriented and consists of a Thing Manager for discovering devices, an Application Manager for storing and returning applications related to the *Things*, and a Data Manager for managing complex queries. The Discovery Service of WoTStore allows the subscription to the device's Thing Description (TD) and notifies the user when the TD is modified. Furthermore, the Thing Manager dynamically creates a Graphical User Interface (GUI) for the TD, thus easing the usage of the devices. In contrast to our approach, WoTStore creates a solution focused on creating a platform for using devices. Our approach aims to facilitate the discovery of CPSs, regarding their technology, by implementing a set of features or capabilities that support the discovery process. However, in our proposal, the modification of the Thing Descriptions is notified in the proactive discovery process. Consequently, users cannot subscribe to the devices registered in the directory, while in WoTStore, users can subscribe and be notified when a Thing Description changes.

DomOS [24] is an approach that implements a Discovery Service based on the recommendation of the W3C. DomOS is an ontology for managing and discovering WoT devices, and authors implement the Discovery Service recommendation of the W3C to evaluate it. Our proposal extends and improves the Discovery Service used to implement DomOS with additional capabilities. Furthermore, the discovery model presented in this paper includes a capability for supporting non-WoT devices.

Other implementations based on the Discovery Service specification of the W3C are WoTHive from the Universidad Politécnica de Madrid, LinkSmart Thing Directory from

Fraunhofer, and LogiLab TDD from Siemens[2]. LinkSmart [40] implements the Discovery Service focused on the Thing Directory. LinkSmart implements a RESTful API for CRUD (Create, Read, Update, Delete), notification, validation, and search operations. Furthermore, the Thing Directory of LinkSmart has authorization and authentication features. Although LinkSmart focuses on implementing a Thing Directory, LinkSmart proposes a service catalogue to discover other web services using HTTP and MQTT.

On the other hand, LogiLab TDD is an emerging implementation developed before the W3C started the recommendation of the Discovery Service for the WoT. Later, Siemens developed SparTDD [14], an evolution of LogiLab that uses the Thing Description architecture proposed by the W3C. SparTDD implements the WoT discovery recommendation that introduces the semantic search using a SPARQL endpoint for searching for TDs.

Finally, WoTHive [9] is the most advanced implementation of the three WoT discovery recommendations. WoTHive research is focused on the discovery feature of the recommendation and implements the Discovery Service recommendation from the W3C using SPARQL and a triple store to register the WoT devices. The discovery is implemented using syntactic and semantic operations. Furthermore, WoTHive compares the performance of syntactic and semantic queries when discovering different amounts of devices. The results show that the semantic search resolves faster than the syntactic search and, in some scenarios, better than the syntactic search. In addition, they propose as future work to research the federation of Discovery Services.

The three implementations of the W3C recommendation of the Discovery Service are based on the published specification [10]. WoTHive is the most developed, introducing the semantic search and proposing the federation of Discovery Services. In contrast to our approach, WoTHive and the LogiLab implementation propose a semantic search of WoT devices using SPARQL. In our approach, the API allows syntactic operations to search for CPSs. Nevertheless, we propose using a recommender system to search using natural language sentences. Our Discovery Service searches for CPSs using syntactic and semantic operations and extends the W3C recommendation by introducing capabilities supporting the discovery process. These capabilities facilitate the discovery of CPSs in different smart environments, thus improving the W3C Discovery Service.

2.2. Background

The discovery model architecture aims to facilitate looking for CPSs in smart environments using web technologies, thus making access to public devices available on the web. The discovery model follows the WoT architecture and uses the TD to store and access CPSs information. In addition, techniques such as pull and push discovery federation systems and artificial intelligence are used to develop the capabilities of the discovery model. Finally, Edge Computing is applied in the topology of the scenario to allow the secure access of external users to CPSs deployed inside a network of different subsystems.

Web of Things. The Web of Things is a framework created in 2010 by Guinard *et al.* [19] to build an ecosystem of the IoT in a flexible, scalable, and open way using web technologies. Lately, the WoT proposal was included as a recommendation of the W3C to define IoT devices that use the Web as the underlying technology [8].

[2] W3C Discovery Services: `https://github.com/w3c/wot-discovery/tree/main/implementations`

For storing the information describing a device in the discovery model, we use the W3C WoT Thing Description, a JSON-LD template document that describes the basic information of WoT devices. The TD provides a model for defining, in a common way, WoT devices to integrate devices and applications, allowing them to interoperate.

The Thing Description defines a set of features called `InteractionAffordances`, which can be properties, actions, or events, and describes how the device interacts and how to interact with it [7]. Properties describe the attributes of the device (e.g., the temperature, actions), actions describe the operations that the device can perform (e.g., turning on a light bulb), and events describe asynchronous notifications that a device may send (e.g., turning on the alarm when the movement sensor detects someone). In our discovery model, `InteractionAffordances` are used by the API for syntactic search and by the recommender system capability for extracting the basic information of each device to train the AI model and recommending devices or services that the user may look for. For instance, searching for temperature sensors or asking for a command to turn on a light.

Pull and push discovery. Apart from adapting the Internet of Things devices into the Web of Things technology [25], the Web of Things requires finding and allowing the secure usage of devices deployed worldwide. Discovery Services solve this problem by facilitating the search for suitable services that meet certain users' requests to solve the problem of the increase in the number of services. Services were increasing, and techniques were needed to perform more efficient queries in the search process [35]. As CPSs are also increasing, there is a need for techniques to facilitate searching for devices. Therefore, Discovery Services can help in the search process for finding devices.

Discovery Services are developed to facilitate searching for services; they can register, unregister and search for services in a directory. Services are inserted into the directory after being registered into the Discovery Service [32]. For registering services, traditional Discovery Services use a push model. The push model follows a reactive behavior, i.e., services or external users register the service; for instance, a user registers a new light bulb in the directory. However, services can also be registered proactively using a pull model, following a more intrusive behavior. This paper proposes a discovery model architecture capable of registering CPSs reactively (push) and proactively (pull). Using a pull model, the Discovery Service looks for deployed services to register them into the directory. To identify available services, the Discovery service, using special bots, crawls the net for services and registers them in the directory. In this sense, the Discovery Service can scan the network or do a DNS-SD search to register all the devices that match specific criteria.

Recommender systems. Recommender systems aim to recommend the products or services that best suit the user's requirements. They are developed to reduce information overload, limiting the information the user gets, thus facilitating the product or service selection. For this reason, recommender systems try to return the most relevant information for the user as the first solution [28].

The recommendation process involves techniques influencing how the recommender system works [29]. Some of the techniques most used for the recommendation process are: (a) Content-based recommendation: Based on products or services similar to those previously selected by the user; (b) Collaborative recommendation: Based on products or

services related to those previously selected by users with similar preferences; (c) Computational intelligence-based recommendation: Focused on building the recommender system using Artificial Intelligence, Bayesian techniques, fuzzy systems, etc.

One of the capabilities of the discovery model architecture is a recommender system based on AI for supporting the search process of CPSs. The reason for using a recommender system is to aid in smart decision-making in large-scale IoT environments, where multiple devices can be the solution, the system may not differentiate between two similar devices, or the user may need help selecting the CPS that best suits the request.

3. Discovery Model Architecture

This paper extends the discovery model proposed in *The 14th International Conference on Management of Digital EcoSystems* [27] by adding a set of capabilities to support the discovery process. As explained in Section 2.1, the proposed discovery model is based on the W3C recommendation of Discovery Service [10] but extended with new features for adapting the Discovery Service to the continuous changes of CPSs.

Figure 1 shows the architecture of the discovery model. The discovery model has four layers for interacting, recommending, processing and storing CPSs. The first layer is the interaction layer, used for interacting with the different layers of the discovery model. The **interaction layer** has a RESTful API deployed using Express.js. The API manages the connections between the discovery model and external agents, working as a security layer. The API has a set of endpoints for communication with the directory to facilitate information retrieval. The new endpoints added to the API implementation are: (a) `#/search`, (b) `#/td/:affordance/:name`, (c) `#created-last-week`, (d) `#updated-last-day`, (e) `#updated-last-week`, and (f) `#user-interface`. In addition, the API has endpoints for interacting with the other three layers. For instance, the API can interact with the processing layer with an endpoint for triggering the proactive discovery capability. Finally, the interaction layer has been extended by including a GraphQL system to help create queries for searching CPS devices. GraphQL allows users to build their queries, increasing the range of queries the API can perform, thus facilitating the search process [34].

The second layer is the **recommender layer**, used for supporting the discovery of CPSs by integrating AI techniques with the discovery model to process natural language queries. This layer is deployed offline using the Transformer algorithm [41] as the base

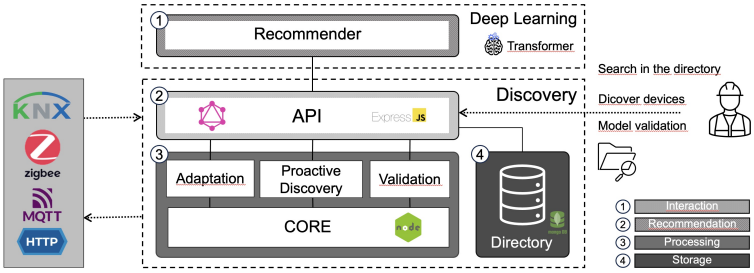

Fig. 1. Discovery model architecture

of the recommender model. In the architecture, this layer is outside the main part of the discovery model architecture to extend it with new features. For instance, a future feature of the recommender layer will be a recommender of Discovery Services to support the federation capability when looking for other Discovery Services in the search process. More in depth-details of the recommender system are explained in Section 3.1.

The third layer is the **processing layer**, which manages the functionality of the architecture. The processing layer uses Node.js and performs all the operations of the discovery model. For instance, creating a paginated list of requested CPSs. These operations are located in the Core of the architecture and are supported by a set of subsystems that extends it; *adaptation*, *proactive discovery*, and *validation*. The adaptation and the proactive discovery subsystems aim to discover WoT devices using a pull model. Furthermore, the adaptation subsystem can also discover CPSs compatible with the TD. In-depth details of these subsystems are described in Section 3.2. Finally, the validation subsystem validates the TD of the CPSs. This subsystem is used to validate the TDs before inserting them into the directory and for validating the dataset used by the recommender system. The TD is validated using the JSON template provided by the W3C.

The fourth layer is the **storage layer**, which stores the CPS information in a MongoDB database following a JSON schema proposed by the W3C. CPSs are stored using the TD and accessed using the identifier of the TD, an immutable field. The stored information is validated by the validation subsystem before inserting them into the directory, thus ensuring that all devices are defined following the same structure. The storage layer can be located outside the main part of the discovery model architecture. Locating the directory outside the main part led to an Edge Computing approach. In an Edge Computing approach, the processing layer is separated from the processing layer, thus adapting the discovery service architecture to federation approaches where directories are located in different subsystems and to topologies in which, for security or network reasons, users cannot access directly the subsystem where devices and the directory are located.

The four layers include a set of capabilities for extending and supporting the discovery model in the discovery process of CPSs. The four proposed capabilities are: (1) a recommender system for matching user queries in natural language with CPSs, (2) proactive discovery for automatically locating CPSs deployed in the same network, (3) a federation for connecting Discovery Services located in different subsystems, and (4) query expansion for improving user queries with additional information which can help in the search process. These four capabilities are described below.

3.1. Recommender system

Discovery Services focus on searching for services using queries. Since natural language is not used in the context of service search, Discovery Services use syntactic and semantic search. However, in the context of CPSs and IoT, natural language may need to be included. For instance, in a smart home scenario, the user may be using both hands, thus needing to interact with CPSs using voice commands. Furthermore, the natural language includes syntactic and semantic, thus allowing more complex searches.

The recommender system capability was presented in a previous paper [26], and it uses natural language for matching user queries in the form of natural language sentences with CPSs. This capability was added to the discovery model to adapt it to the new scenarios of CPSs that were not contemplated in the context of the services. As CPSs can be

deployed in scenarios where users without computer engineering knowledge can interact with them, the system must be adapted to be used by that kind of user. Furthermore, using natural language allows interaction by text or voice, facilitating the use of the system for users with disabilities that may need to interact with the system. In addition, as natural language includes syntactic and semantic techniques, queries for searching devices can be sent to the Discovery Service or the recommender, thus having two ways of searching for devices in the Discovery Service.

The recommender system is proposed as a support system because it needs to be trained when new CPSs are added to the scenario. Therefore, it cannot be used for searching for new CPSs. While the recommender learns from the new CPSs, the Discovery Service API is used together with GraphQL as the search system. The recommender system is built using deep learning through a Transformer approach, a novel sequence transduction model based on multi-head self-attention [41] that presents good results in natural language problems. A user sends a natural language sentence, and the recommender returns the four devices that best suit the user's request. The returned list is sorted by the recommender's confidence about each device being the solution to the user's request.

This capability is being improved to recommend CPSs services, e.g., return the command needed to turn on the light bulb in the kitchen. In addition, this capability will be extended to support recommendations in a Discovery Service federation for CPSs.

3.2. Proactive discovery

Discovering services includes searching for them in a directory and registering the available services in the directory. For registering, services must be inserted manually into the directory, either by the services registering themselves or by an external user registering them. Therefore, services must be adapted to know the presence of the Discovery Service, or an external user has to spend time registering the services.

In the context of CPSs, one of the features of IoT devices is their ability to be dynamic regarding the location where they are deployed. A car, a scooter, or a bike can be CPSs with mobile capabilities that change their location continuously. Regarding less complex scenarios, a laboratory where devices are added and removed every day can also be an example of the non-static feature of CPSs. Therefore, Discovery Services have to be adapted to discover dynamic cyber-physical devices.

The proactive discovery subsystem was initially presented in [27], and extended in this paper. To adapt to the non-static feature of CPSs, the proactive discovery subsystem allows the discovery of CPSs by following a pull model. The Discovery Service scans the subnet, searching for WoT devices. As WoT devices that use the HTTP communication protocol deploy their Thing Description document in port `80`, the subsystem searches for systems with port 80 opened and a TD deployed in the root path. After devices are located, TDs are registered in the directory.

The proposed proactive discovery subsystem facilitated the discovery of WoT devices using the HTTP protocol and deployed them in the same subnetwork as the discovery. However, this solution cannot discover IoT or WoT devices that use other communication protocols, such as MQTT, Zigbee, or KNX. In [25], the discovery model was extended with a new subsystem for adapting IoT devices to the WoT technology, thus proactive discovering IoT and WoT devices. As adapting IoT devices to WoT technology can be difficult, the adapt feature was developed to work with a more structural communication

protocol than HTTP, the MQTT protocol. Therefore, the adaptation subsystem can only adapt IoT devices using the MQTT communication protocol to the WoT technology. Further information about the adaptation process is described in [25]. Finally, the proactive discovery subsystem was extended to discover WoT devices using MQTT. The proactive discovery subsystem can discover WoT devices using HTTP or MQTT and IoT devices using MQTT. For discovering devices using MQTT, the subsystem looks for devices deployed in the same network with the ports `1883`, and `8883` opened.

The adaptation subsystem is linked with the proactive discovery subsystem, i.e., the only way to execute the adaptation operation is by triggering the proactive discovery. The proactive discovery can be executed manually or automatically. The automatic approach executes the proactive discovery operation, triggering the network scan when an event happens. For instance, a timed event triggers the network scan each hour.

The manual approach executes the proactive discovery when an external user triggers the operation. The API has an endpoint for interacting with the proactive subsystem, thus allowing external users to trigger it. When a user executes the proactive discovery, the new CPSs detected are not registered in the directory. The CPSs discovered are returned to the user using three subsets, one subset for the new CPSs, one subset for the CPSs already registered in the directory, and one subset for CPSs that have changed since they were discovered. Therefore, users can decide what to do which each subset. For instance, users can be interested only in CPSs that have changed since the last time they were discovered.

This capability is currently being investigated to support more communication protocols. In addition, because the subnetwork scanning works in a broadcast form, it will be improved to avoid overloading the network, thus being less intrusive, and to solve security problems found when deploying the experimental scenario. Finally, the proactive discovery will be improved to adapt IoT devices using other communication protocols and solve the problem when searching for sleeping devices, i.e., devices that deactivate features that, after some time without being used, reduce energy consumption and limit incoming connections.

3.3. Federation

In the context of Cyber-Physical Systems, users must interact with devices in different subsystems. For instance, using devices located on different floors inside the same building. In some situations, Discovery Services can be deployed to discover all these devices. However, the connection range and security may not allow a single Discovery Service to discover all the available CPSs. Therefore, a set of Discovery Services must work together to discover CPSs deployed in different subsystems.

The federation capability is proposed to connect CPSs in different subsystems, making it easier to discover them. Discovery Services know other Discovery Services, thus looking for CPS devices in other directories when not located locally.

Discovery Services have maximum hops in a federated discovery process to limit the search and reduce the waiting time. For instance, Figure 2 shows a federated Discovery Service; If the hop limit is set to one for the Discovery Service #1, it will only search in those Discovery Services located in Level 1, i.e., #2 and #3.

Another feature of the federation capability is the confidence level of each Discovery Service. Discovery Services have a list of other Discovery Services to delegate queries to them when the CPS is not in their directory. To sort that list, the Discovery Service

gives a confidence value to each Discovery Service based on features such as the number of times the Discovery Service returns the correct device, the security, or reliance, among others. The delegation of queries inside a federation is a feature that could become another capability if it is developed enough to be used as a separate system.

Finally, the recommendation capability works with the federation capability to allow the recommendation of CPS inside each Discovery Service and to extend this feature with the recommendation of Discovery Services in the federation. The recommender works with the confidence value given to each Discovery Service to recommend a Discovery Service over others. For instance, in Figure 2, the recommendation system may recommend to the Discovery Service #1 the query delegation to Discovery Services #2 and #5.

The federation capability is currently an ongoing research. One of the difficulties in finishing this proposal is to propose a discovery model federation that can delegate queries while minimizing the waiting times. For instance, if a query is delegated by sending it one by one to each Discovery Service, the waiting time is the sum of all waiting times. However, if the query is delegated by sending it to all the Discovery Services simultaneously, the waiting time will be the waiting time of the slowest Discovery Service.

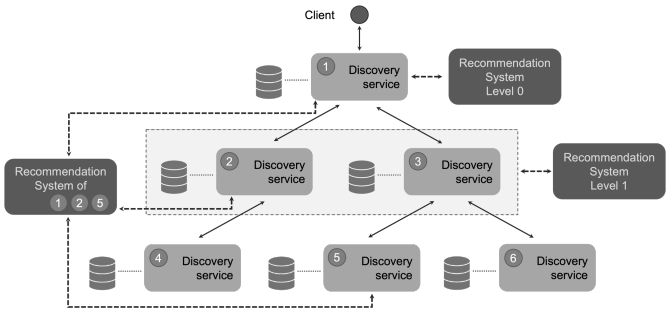

Fig. 2. Federation of Discovery Services

3.4. Query expansion

Searching for CPSs may differ from searching services in the way of the kind of users that interact with the system and how they interact with it. In the research process of the recommender system, we detected that users might not know how to ask for what they need or not include all the required information in the query. For instance, a user wants to turn on the light bulb in the living room and sends a query in the form of a natural language sentence that says: *Turn on the light*. Therefore, the system cannot know which light bulb has to be turned on.

The last proposed capability for the discovery model is **query expansion**, a technique for extending user queries with additional information or reformulating the query to enhance the information retrieval effectiveness [2], [6]. This capability is proposed to support the recommender system by improving the user queries to include information that can help the recommendation process. As natural language sentences can be abstract,

the query expansion system extracts information such as the user's location to extend the query. With abstract sentences, we mean sentences that are not built correctly or sentences that do not specify the user's desired device. For instance, *Turn on the light* can mean illuminating a whole room, but the system does not know the location of the lights, and the room may have more than one light. This sentence can be modified into *Turn on the lights in the living room*, thus improving the recommendations of the recommender system.

Finally, the query expansion capability may be useful for searching devices by reformulating users' queries. After a query is resolved without results, the query is reformulated to get any device in the search result that may suit the user's request. The query expansion capability is being developed. As it may interact with the recommender and the Discovery Service search process and it has to extract information from the user, it will be located in the main Discovery Model as a subsystem. The challenges for this capability are about what information to extract, how to do it, and how to reformulate queries to improve the result without returning devices that the user may not need.

4. Experimental Scenario

In IoT, no public scenarios exist to experiment and interact with CPS. The existing scenarios, such as FIT IoT Lab [1], are limited or with restricted access. Regarding the WoT, there are three available scenarios, Remote Lab from the Technical University of Munich [33], a public scenario from the book *Building Web of Things* [18], and IoTLab [3]. This WoT scenario does not follow the W3C recommendation. However, these laboratories require access rights or are very limited. For instance, IoTLab only uses CoAP devices and does not use the TD, and [18] has only two devices available.

For the experimental scenario, a Discovery Service implementation of the discovery model is deployed in a public laboratory. The laboratory has three rooms connected to a local network inside one of the subnetworks of the institution. This scenario helps researchers experiment with WoT devices of different smart environments and providers. Furthermore, other scenarios can link to our laboratory, thus helping create a federation of Discovery Services.

In this section, first, the public laboratory, the WoT Lab[3], is described. After describing the WoT Lab, a scenario of experimentation and how users interact with the scenario using the Discovery Service is explained.

4.1. WoT Lab

WoT Lab is a public environment where users can experiment with CPSs in different smart scenarios. Users can interact with devices using the user interface or by asking directly to the Discovery Service. Each device has a Thing Description and the endpoints required for interacting with them. For instance, a smart suitcase simulates devices from a Smart Home; the website publishes the Thing Description of the suitcase, and the endpoints for interacting with each device from the suitcase are available on the Thing Description.

Figure 3 shows the architecture of the WoT Lab. Devices are deployed in three different rooms connected to the same local network. Some devices use different communication protocols, including KNX, HTTP, MQTT, and Zigbee. Furthermore, devices are

[3] WoT Lab website: `https://acg.ual.es/projects/cosmart/wot-lab/`

from different brands to experiment with interoperability. For instance, there are motion sensors from Bosch, KNX, and Philips; and light bulbs from Philips and Ikea.

The website, the Discovery Service, and the directory of the Discovery Service are deployed inside a docker container to facilitate their management and deployment of a federation of Discovery Services. Discovery Services can be deployed in containers in different networks and connected to allow the discovery of CPSs in different subsystems.

The Discovery Service and the website must access the deployed devices' network to interact with the devices. As external users have to access the website without access to the internal network and the website must have access to the internal network, it could cause security problems described in the next subsection. Users ask the website for CPSs; the website asks the Discovery Service for the CPSs, and the Discovery Service looks for them in the internal network and the directory.

4.2. Scenario

The experimental scenario describes the interaction of the Discovery Service and its capabilities with the IoT ecosystem. In addition, it shows how users can access and interact with the CPSs through the Discovery Service.

The experimental scenario is based on the WoT Lab approach and has devices in three rooms connected to a local network deployed in one of the rooms (Figure 4). In the local network (LN1), devices can communicate between them, and the Discovery Service can use the proactive capability to search for them by scanning the network. The available devices are (a) a smart suitcase that simulates a Smart Home, (b) three light bulbs, (c) three motion sensors, (d) four contact sensors deployed in two windows and two doors, (e) a video camera, (f) three temperature and humidity sensors, (g) three CO_2 and temperature sensors, and (h) two servos for interacting with the system. All these devices are registered in the Discovery Service directory and are available in the WoT Lab.

To access the Discovery Service, thus interacting with devices, users must connect to the subnetwork N1 of the rooms' building. As N1 can access to LN1, and the Discovery Service must be in LN1 to discover the devices, the Discovery Service is deployed in LN1. The Discovery Service scans the local network once each hour by triggering the proactive discovery capability and registers the new devices deployed. Figure 5 shows a sequence diagram of the interaction of the Discovery Service when making a proactive discovery of two new devices, a smart mailbox that uses MQTT and a smart blind that uses HTTP under the WoT technology. First, the Discovery Service scans the network looking for

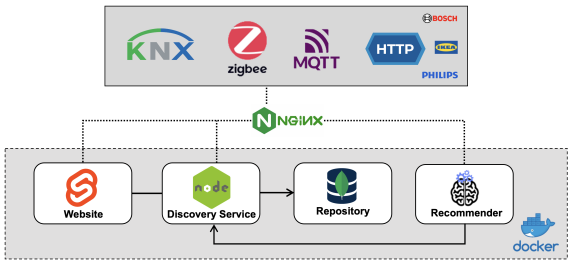

Fig. 3. WoT Lab architecture

HTTP devices; it can discover devices already registered, devices registered whose TD has been modified, and new devices. In this scenario, a new device, the smart blind, is discovered. After detecting HTTP devices, the Discovery Service scans the network for MQTT devices. In this example, one IoT device, the mailbox, is detected. Before registering both devices into the directory, the Discovery Service calls the adaptation subsystem to generate a TD for the mailbox. Finally, the Discovery Service calls the validation subsystem to validate the TD of the smart blind and registers both devices into the directory. In the case of an error when validating the smart blind's TD, the blind is not registered.

The average time of the proactive discovery to scan and register devices in the network of the experimental scenario is 138 seconds. In contrast, the average time for resolving the queries for discovering and registering the devices of the example is 207 milliseconds. The proactive discovery time is higher than it would take to register a device manually. However, proactive discovery does not require human intervention. Therefore, it is lower than the time required to adapt IoT devices to WoT technology or to the time required to register a large number of IoT devices. Furthermore, suppose the network cannot handle a continuous broadcast. In that case, the proactive discovery subsystem can be disconnected and only triggered by querying an endpoint available in the API of the Discovery Service, e.g., calling the endpoint when more than ten new devices have been deployed. This capability can help in scenarios where the user does not know how the devices work or the number of features of the device. For instance, the CO_2 sensors in the three rooms have additional features that were not documented. Using the proactive discovery and adaptation subsystem to discover and register these devices helped to identify their features.

The Discovery Service must be connected to the directory to search for devices. In this scenario, the directory and the Discovery Service are deployed on LN1, thus having bidirectional communication. Users can search for devices using the website, which calls some of the endpoints of the API, or they can directly send the queries to the Discovery Service. For instance, users can see the current online devices on the website and main page. For showing the online devices, the website queries the Discovery Service to list all the devices with the status online in the Thing Description. In addition, users can build their queries using the GraphQL server deployed in the Discovery Service[4]. GraphQL uses the JSON schema file of the Thing Description for structuring the information and helping users in the building process of the queries. However, these search techniques

[4] GraphQL subsystem: `https://acg.ual.es/projects/cosmart/wot-lab/ds/graphql/`

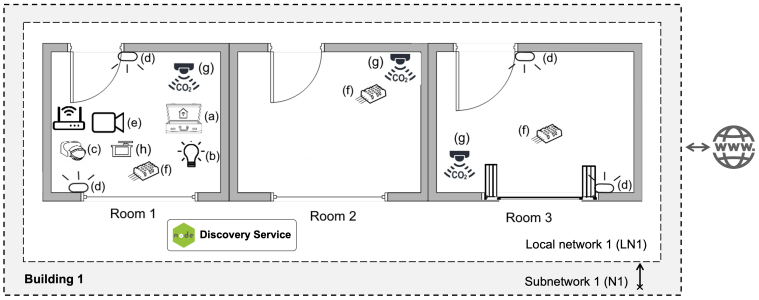

Fig. 4. Experimental scenario

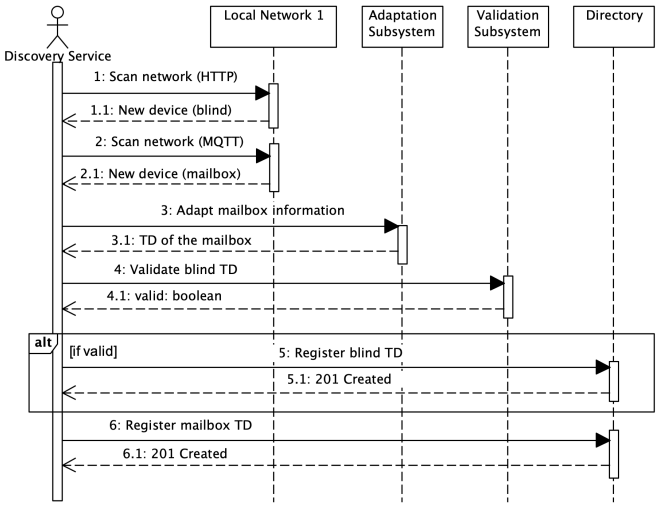

Fig. 5. Sequence diagram of the proactive discovery

cannot process natural language sentences and cannot sort the result of the search query by a value that can help the user make decisions. A user may not know which device wants when searching for it. For instance, in our scenario, a user may want to increase the illumination in the room, but the user may not know that one of the light bulbs is broken and that by opening the blind, the illumination can be increased. Therefore, a recommendation in the query result may help the user decide by returning the blind at the top and the broken light bulb at the bottom of the list.

As the recommender system is a capability that supports the search process of CPSs, it must be connected to the Discovery Service, thus being deployed in LN1. The recommender is deployed in a docker container with the Discovery Service and the website. The dataset used to train the recommender to recommend the deployed CPSs is the dataset from [26]. The dataset was adapted to return the name of each device instead of the generic names of the dataset. For instance, when searching for light bulbs, the available light bulbs are *WoTColorLight1*, *WoTColorLight3*, and *WoTColorLight4*. Since the dataset is limited in the data used for training it, the recommender cannot differentiate between devices of the same type, thus returning all the lights of each room when searching for them. For instance, if the user searches for a specific light bulb in *room1*, the recommender system will return the three light bulbs deployed in *room1*. For the training process, the recommender system accesses the Discovery Service validation subsystem to validate the dataset used for training the model. However, the recommender system can not re-train itself with the information collected when using the recommender system or with the queries sent to the Discovery Service. After training, the recommender system is deployed in LN1, thus allowing external access to the service[5].

As explained, users may not know the device they want. Furthermore, users may not include all the information required to find their desired CPS. The query expansion ca-

[5] RS: `https://acg.ual.es/projects/cosmart/wot-lab/transformer/predict/Turnonthelight`

pability supports the search and recommendation process by completing the queries with additional information and reformulating them. Query expansion is under research, but some experiments were carried out in this scenario as a supporter of the recommender system. In the adaptation process of the dataset, some of the sentences were modified, and new sentences were included. The reformulation of these sentences followed a structure to include the required information for helping the recommender system in the recommendation process (1) (2). As this is the first approach of the query expansion capability, the structure followed for building the sentences is not complex. In the proposed structure, a sentence is built by adding information about the type of device wanted (D), the location of the device (L), and the action desired (A) (3). The action of the device is included because of future Artificial Intelligence models where the recommender system returns a CPS's service instead of a specific device. For instance, the user wants to turn on a light bulb, and the recommender system returns the endpoint used for turning the light bulb on.

$$A = \{a_1, a_2, ..., a_n\}; D = \{d_1, d_2, ..., d_n\}; L = \{l_1, l_2, ..., l_n\} \tag{1}$$

$$a_1 \cup d_1 \cup l_1 \tag{2}$$

Turn on light bulb room 1. (3)

More complex sentences that can be built using the proposed structure involve more than one action, device, or location (1). A simple structure is used because the recommender system cannot process these kinds of sentences.

To access the experimental scenario in LN1, external users must connect to N1. However, the connection between N1 and external networks was restricted for security reasons. Users can now only access N2, a network isolated from all the networks. Therefore, after this change, users could not access the experimental scenario. In the new topology, N1 can still connect to LN1 and N2. However, external users cannot access N1. Furthermore, external users can connect to N2, but N2 cannot connect to N1. To solve the problem in the connection between external users and the experimental scenario, the Discovery Service was deployed following a first approach of a federation of Discovery Services.

The Discovery Service deployed in LN1 continues to work in LN1 to discover the devices deployed in the network. The topology modification focuses on the communication between N1 and N2, allowing external access to the experimental scenario. The topology modification follows a federation approach, where an additional Discovery Service is deployed in N2. External users can access the Discovery Service to access devices in LN1. As N1 can access N2, the Discovery Service in N1 is connected with the Discovery Service in N2, updating the information stored in the directory of N2 with the information of the devices in LN1. Furthermore, as there are no devices in N2, the proactive discovery of the Discovery Service in N2 is disabled to avoid overloading.

The proposed discovery model works as a federation of Discovery Services, where more Discovery Services can be deployed in other networks and connected to the Discovery Service deployed in N2. This scenario is the first approach to the capability of the federation. However, it is still under research, in this experimental scenario, the failure of the Discovery Service of N2 would make all the Discovery Services to disconnect from the external network.

5. Discussion and Threats to validity

A discovery model is proposed and deployed in a scenario presented in [27] and extended with more devices in three rooms. The discovery model consists of multiple capabilities to adapt Discovery Services to the evolution of CPSs. For instance, in a Smart Home scenario, the Discovery Service can facilitate the discovery of new devices deployed by automatically discovering them, thus helping people without technical experience to manage the CPSs of the Smart Home. The recommender system and query expansion capabilities help process natural language sentences sent by the user when using the CPSs. Finally, the federation capability helps in the search process of CPSs deployed in different subnetworks of the Smart Home scenario.

In other scenarios, such as Smart City, the most relevant capability is the federation. As a city is too large for a single Discovery Service, multiple Discovery Services must be deployed in the areas of the city to manage all the CPSs. The Discovery Services are connected to the federation to create a way of searching efficiently for devices in the city.

As shown in the scenarios, the proposed capabilities support the discovery model, adapting it to different smart environments. However, the discovery model proposal is still under research and suffers from limitations and threats to validity. Therefore, to validate our proposal, we answer the four main validity threads discussed in the literature [13]: Conclusion, Internal, Construct, and External validity. This ensures detection of the objectives are fulfilled and the study's limitations.

Conclusion validity. Did the introduced treatment/change have a statistically significant effect on the outcome we measure? Yes, the results obtained using the proposed discovery model differ from those obtained using other Discovery Service approaches. The federation, recommender system, and query expansion capabilities modify the returned list of CPSs. The federation returns devices from the current Discovery Service and other Discovery Services linked to it. The recommender system modifies the output by returning the four devices that best suit the user's request. Finally, the query expansion reformulates the query to improve the search result.

These three capabilities make the output of our proposal differ from the output of other Discovery Services that search in the same scenario. However, there is a lack of comparison between our proposal and other Discovery Services in the literature. For instance, comparing our discovery model with the Discovery Services implementations of the W3C recommendation explained in the Related Work of Section 2.1.

A useful comparison between our approach and other proposals is the precision and recall of the output when searching for devices. This can help measure the output's improvement using the proposed capabilities. Furthermore, the capabilities may slow the response time of the Discovery Service. Therefore, comparing the performance between our Discovery Service and other approaches may be useful.

Internal validity. Did the introduced treatment/change cause an effect on the outcome? Can other factors also have had an effect? The outcome is altered as we use the capabilities to support the discovery process. The discovery result from not using the capabilities differs from the result from using them. In addition, the proactive discovery may alter the devices available in the directory, thus altering the outcome. However, the proactive discovery and recommender capabilities may slow the query time.

In the experimental scenario, we compared the average time of proactive discovery and manually discovering and registering the devices. However, this comparison is not enough to evaluate the capability. Furthermore, the response time between the recommender and the API is not compared. Therefore, the response time of the proactive discovery and the recommender should be studied in-depth. For instance, the proactive discovery must be compared with the manual approach using a different amount of devices.

Construct validity. Does the treatment correspond to the actual cause we are interested in? Does the outcome correspond to the effect we are interested in? The research aims to: (a) propose a discovery model for the WoT that improves and complements the Discovery Service proposed in the W3C recommendation; (b) propose a discovery model that facilitates the discovery of CPSs; and (c) create a public laboratory for the WoT.

Objectives (a) and (b) are fulfilled in Section 3, compared with other approaches in Section 2.1, and studied in Section 4; and objective (c) is fulfilled in Section 4. For objective (a), the questions related to the proposal of a discovery model that complements the recommended by the W3C have been answered affirmatively. However, other questions related to improving the Discovery Service recommended by the W3C were not answered affirmatively. The proposed discovery model has more features than the Discovery Service proposed by the W3C. However, to assert that our approach improves the W3C recommendation, they must be compared using metrics from the literature to evaluate the difference between both approaches.

External validity. Is the cause-and-effect relationship we have shown valid in other situations? Can we generalize our results? Do the results apply in other contexts? The discovery model is deployed in implementing a Discovery Service in the proposed scenario, the WoT Lab. WoT Lab consists of a set of devices from different smart environments to help research the use of CPSs in these devices. Although the number of devices has been increased from the paper, which it extends for, the number of devices is too small for generalizing the usage of the discovery model in all the smart environments. Therefore, the number of devices used in the WoT Lab must be increased. The discovery model must be validated in another scenario to compare both experiments and show that the proposal is valid in other situations.

6. Conclusions

This paper proposes a discovery model architecture for Cyber-Physical systems based on the Web of Things. In particular, the proposed mechanism is intended to store, search and facilitate access to devices represented and controlled under the W3C Web of Things paradigm (commonly referred to as WoT). Furthermore, the presented approach extends previous work by including different capabilities, such as proactive discovery, recommendation, and federation, to facilitate the discovery of CPSs.

A public laboratory, the WoT Lab, was deployed to analyze the discovery model. The WoT Lab is a laboratory presented before and extended in this paper by increasing the number of devices. In addition. a Discovery Service was implemented in the laboratory following the discovery model architecture. As proposed in the previous work, the experimental scenario was modified to improve the security of the approach, using Edge

Computing facilities to interact and manage CPSs. Finally, each capability was validated in the scenario to show how they can facilitate the discovery of CPSs, thus affirmatively answering the second research question.

Regarding the first research question, other implementations of Discovery Services in IoT and WoT were studied in Section 2.1. Furthermore, in Section 3 and 4, the relevance of extending the Discovery Service to adapt to the CPSs was explained and validated, thus answering the first research question affirmatively.

Future work could improve the validation by comparing our proposal with others, such as WoTHive and LinkSmart Thing Directory. Furthermore, the capabilities could be compared to an approach that not uses them to evaluate the performance of a more complex Discovery Service against more traditional ones. Finally, we intend to increase the number of devices in the experimental scenario and include another scenario in the validation to compare the proposal's performance in both and generalize our solution.

Acknowledgments. Grant PID2021-124124OB-I00 funded by MCIN/AEI/10.13039/501100011033 and by the "ERDF A way of making Europe". Grant PY20_00809 funded by the Andalusian Government and the "European Union". Grant FPU19/00727 funded by MIU and the Spanish Government.

References

1. Adjih, C., Baccelli, E., et al.: FIT IoT-LAB: A large scale open experimental IoT testbed. In Proc. of IEEE WFIoT'2015, 459–464. (2015)
2. Azad, H. K., Deepak, A.: Query expansion techniques for information retrieval: A survey. Information Processing & Management, Vol. 56, No. 5, 1698-1735. (2019)
3. Belli, L., Cirani, S., et al.: Design and Deployment of an IoT Application-Oriented Testbed. Computer, Vol. 48, No. 9, 32-40. (2015)
4. Borozdukhin, A., Dolinina, O., Pechenkin, V.: Approach to the garbage collection in the "Smart Clean City" project. In Proc. of IEEE CiSt'2016, 918-922. (2016)
5. Campioni, L., Fontana, N., et al.: A Federated Platform to Support IoT Discovery in Smart Cities and HADR Scenarios. In Proc. of FedCSIS'2020, 511–519. (2020)
6. Carpineto, C., Romano, G.: A Survey of Automatic Query Expansion in Information Retrieval. ACM Comput. Surv., Vol. 44, No. 1, 1-50. (2012)
7. Charpenay, V., Käbisch, S.: On Modeling the Physical World as a Collection of Things: The W3C Thing Description Ontology. In Proc. of ESWC'2020, 599–615. (2020)
8. Charpenay, V., Käbisch, S., Kosch, H.: Introducing Thing Descriptions and Interactions: An Ontology for the Web of Things. In Proc. of IoT'2016, 55–66. (2016)
9. Cimmino, A., García-Castro, R.: WoTHive: Enabling Syntactic and Semantic Discovery in the Web of Things. Open Journal of Internet Of Things, Vol. 8, No. 1, 54–65. (2022)
10. Cimmino, A., McCool, M., Tavakolizadeh, F., Toumura, K.: Web of Things (WoT) Discovery, W3C Candidate Recommendation Snapshot 19 January 2023. (2023)
11. Criado, J., Iribarne, L.: Reusability and discovery models in software systems: a systematic literature review. Journal of Object Technology, Vol. 21, No. 4, 1–17. (2022)
12. EPCglobal.: GS1, Discovery Configuration and Initialization (DCI). (2023) [Online]. Available: https://www.gs1.org/standards/epc-rfid/dci/1
13. Feldt, R., & Magazinius, A.: Validity Threats in Empirical Software Engineering Research - An Initial Survey. In Proc. of SEKE'2010, 374-379. (2010)
14. Glomb, C., Thiéblin, É., Amarger, F.: SparTDD-a SPARQL based Thing Description Directory. In Proc. of ESWC'2022, 1-6 (2022)

15. Gomes, P., Cavalcante, E., et al.: A semantic-based discovery service for the Internet of Things. Journal of Internet Services and Applications, Vol. 10, No. 10., 1–14 (2019)
16. Gomes, P., Cavalcante, E., et al.: A Federated Discovery Service for the Internet of Things. In Proc. of M4IoT'2015, 25-30. (2015)
17. Greer, C., Burns, M., Wollman, D., Griffor, E.: Cyber-physical systems and internet of things. NIST Special Publication 1900–202. (2019)
18. Guinard, D., Trifa, V.: Building the Web of Things. Manning Publications. (2016)
19. Guinard, D., Trifa, V., Wilde, E.: A resource oriented architecture for the web of things. In Proc. of IoT'2010, 1–8. (2010)
20. Iggena, T., Bin Ilyas, E., et al.: IoTCrawler: Challenges and Solutions for Searching the Internet of Things. Sensors, Vol. 21, No. 5, 1559. (2021)
21. Iribarne, L., Troya, J. M., Vallecillo, A.: A Trading Service for COTS Components. The Computer Journal, 47(3), 342–357. (2004)
22. Kabalci, Y., Kabalci, E., et al.: Internet of Things Applications as Energy Internet in Smart Grids and Smart Environments. Electronics, 8(9). (2019)
23. Kovatsch, M., Matsukura, R., et al.: Web of Things (WoT) Architecture, W3C Recommendation. (2023) https://www.w3.org/TR/2020/REC-wot-architecture-20200409/
24. Laadhar, A., Thomsen, C., Pedersen, T. B.: domOS Common Ontology: Web of Things Discovery in Smart Buildings. The Semantic Web: ESWC 2022 Satellite Events, 95–100.
25. Llopis, J. A., Criado, J., Iribarne, L., Padilla, N.: A Discovery Pull Model for Devices in IoT and WoT Environments. In Proc. of IoT'2021, 228–233. (2021)
26. Llopis, J. A., Fernández-García, A. J., Criado, J., Iribarne, L.: Matching user queries in natural language with Cyber-Physical Systems using deep learning through a Transformer approach. In Proc. of INISTA'2022, 1–6. (2022)
27. Llopis, J. A., Mena, M., Criado, J., Iribarne, L., Corral, A.: Towards a Discovery Model for the Web of Things. In Proc. MEDES'2022, 96–103. (2022)
28. Lu, J., Wu, D., Mao, M., Wang, W., Zhang, G.: Recommender system application developments: A survey. Decision Support Systems, 74, 12–32. (2015)
29. Lü, L., Medo, M., Yeung, et al.: Recommender systems. Physics Reports, 519(1), 1–49. (2012)
30. Marwedel, P.: Embedded system design: embedded systems foundations of cyber-physical systems, and the internet of things. Springer Nature. (2021)
31. Mayer, S., Guinard, D.: An extensible discovery service for smart things. Proceedings of the Second International Workshop on Web of Things, 1–6. (2011)
32. Memon, S., Jens, J., Willem, et al.: Towards Federated Service Discovery and Identity Management in Collaborative Data and Compute Cloud Infrastructures. Journal of Grid Computing, 16(4), 663–681. (2018)
33. Remote Lab. Tech. Univ. of Munich, Germany. (2023) https://esiremotelab.esi.ei.tum.de
34. Porcello, E., Banks, A.: Learning GraphQL: declarative data fetching for modern web apps. O'Reilly Media, Inc. (2018)
35. Pourghebleh, B., Hayyolalam, V., Aghaei, A.: Service discovery in the Internet of Things: review of current trends and research challenges. Wireless Networks, 26(7), 5371–5391. (2020)
36. Rajkumar, R., Lee, I., Sha, L., Stankovic, J.: Cyber-physical systems: The next computing revolution. Design Automation Conference, 731–736. (2010)
37. Sani, M. F., et al.: Improving the performance of process discovery algorithms by instance selection. Computer Science and Information Systems, 17(3), 927–958. (2020)
38. Sciullo, L., Gigli, L., Trotta, A., Di Felice, M.: WoT Store: Managing resources and applications on the web of things. Internet of Things, 9. (2020)
39. Shi, J., Wan, J., Yan, H., Suo, H.: A survey of Cyber-Physical Systems. In Proc. of WCSP'2011, 1–6. (2011)
40. Tavakolizadeh, F., Devasya, S.: Thing Directory: Simple and lightweight registry of IoT device metadata. Journal of Open Source Software, 6(60), 3075. (2021)

41. Vaswani, A., Shazeer, N., Parmar, N., et al.: Attention Is All You Need. In Proc. of NIPS'2017, 6000–6010. (2017)

Juan Alberto Llopis is a Phd Student researching, under the national project TIN2017-83964-R, about Federation service discovery in the Web of Things after receiving a FPU grant (ref. FPU19/00727). His research focuses on: Internet of Things, the Web of Things and Smart Environments. His email address is jalbertollopis@ual.es.

Manel Mena is a PhD in CS at the University of Almeria. Currently he is working in researching IoT Systems, Software Architectures and the Web of Things. From 2018 to 2022, he was supported by an FPU grant (ref. FPU17/02010). His interests include Data Engineering, Software Engineering, Cloud Computing, Microservice Architectures, Component-Based Software Engineering, Machine Learning, the Web of Things and Model-Driven Engineering. His email address is manel.mena@ual.es.

Javier Criado is an Associate Professor at the Department of Informatics and the Applied Computing Group (ACG), University of Almería (Spain). His main research areas are model-driven engineering, component-based software engineering, user interfaces, interoperability, service-oriented architectures, microservices and web of things. You can contact the author at javi.criado@ual.es.

Luis Iribarne is a Full Professor at the Department of Informatics, University of Almeria (Spain), and Head of Applied Computing Group. He has served as the Principal Investigator for nine R&D projects founded by the Spanish Ministry of Science and Technology. His main research interests include simulation, component-based software development, model-driven engineering, and software engineering. His email address is luis.iribarne@ual.es.

Antonio Corral is an Associate Professor at the Department of Informatics, University of Almeria (Spain. He has participated actively in several research projects in Spain and Greece. His main research interests include access methods, algorithms, query processing, spatial databases and distributed query processing. His email address is acorral@ual.es.

Received: March 28, 2023; Accepted: July 01, 2023.