


# ODRL-Based Provisioning of Thing Artifacts for IoT Applications

Zakaria Maamar<sup>1</sup><sup>a</sup>, Amel Benna<sup>2</sup><sup>b</sup> and Haroune Kechaoui<sup>3</sup>

<sup>1</sup>University of Doha for Science and Technology, Doha, Qatar

<sup>2</sup>Research Center for Scientific and Technical Information (CERIST), Algiers, Algeria

<sup>3</sup>High School of Computer Science, Algiers, Algeria

**Keywords:** Composition, Discovery, IoT, Policy, ODRL, Thing Artifact.

**Abstract:** This paper discusses the design and development of Internet-of-Things (IoT) applications based on the novel concept of Thing Artifact (TA). A TA is multi-faceted having a functionality, life cycle, and interaction flows. Prior to integrating TAs into an IoT application, they need to be discovered and then, composed. While existing discovery and composition techniques are functionality-driven, only, this paper demonstrates that policies regulating the functioning of TAs in terms of what they are permitted to do, are prohibited from doing, and must do, have an impact on their discovery and composition. These policies are specified in Open Digital Rights Language (ODRL). A system implementing and evaluating ODRL-based provisioning of TAs for IoT applications design and development is presented in the paper, as well.


## 1 INTRODUCTION


To address the complexity of Internet-of-Things (IoT) applications design and development, we presented in (Maamar et al., 2023) the concept of Thing Artifact (TA). Contrarily to things that are the building blocks of these applications, a TA leverages the concept of Data Artifact (DA) that is commonly used to manage data-driven business applications (Nigam and Caswell, 2003). Because of this leveraging, we provide a rich description of a TA using 3 cross-cutting aspects: functionality in term of what a TA does; life-cycle in term of how a TA behaves; and interaction flow in term of with whom a TA exchanges messages. For illustration, we abstracted a smart home in (Maamar et al., 2023) with multiple collaborative TAs like smartTV:TA, remoteControl:TA, and motion-Detector:TA provisioning different functionalities, exposing different lifecycles, and participating in different interaction flows.

For a successful buy-in of TAs by the information and communication technology community, in general, and IoT community, in particular, techniques to define, discover, and compose TAs are necessary. A first step is to adopt and, most probably, adapt existing thing techniques like the IoT composer by Kr-

ishna et al. (Krishna et al., 2019) and ComPOS by Åkesson et al. (Åkesson et al., 2019) (Section 2.3). However, these techniques overlook the compliance of TAs with organizations' business rules. A TA that responds to invocation requests between 10am and 11am, only, cannot be integrated into an IoT application that submits such requests at any time of the day. The invocation time-period that the IoT application imposes on the TA is a decisive factor in its selection. TAs discovery should go beyond their functionalities by having extra factors like time, location, and budget.

In this paper, our TAs discovery and composition techniques resort to Open Digital Rights Language (ODRL, (Cano-Benito et al., 2023; W3C, 2018)) to guarantee that TAs' future uses do not jeopardize IoT applications. This happens by examining TAs' ODRL policies during discovery and composition. A TA that is permitted of sensing during an ODRL-set time period cannot sense outside this time period if requested by an IoT application. And, a TA that is prohibited from communicating some users' location information can still communicate other information to an IoT application. In either case, the rationale of why, when, where, and how TAs will be used impact their discovery and composition; sensing is permitted in association with a specific time period and communicating is prohibited in association with some sensitive information. How to enforce these as-

<sup>a</sup> <https://orcid.org/0000-0003-4462-8337>

<sup>b</sup> <https://orcid.org/0000-0002-9076-5001>

sociations is a question to address.

Being a rights expression language, ODRL policies consist of permission, prohibition, and duty rules that control the use of assets like data, service, and media. ODRL expresses that “*something is permitted, forbidden, or obliged, possibly limited by some constraints*” (W3C, 2018). Contrarily to assets that are static in ODRL, our TAs are dynamic as per their life-cycles requiring a dynamic enactment of rules. For instance, a permission rule for a smartTV:TA is enacted when this TA takes on the activated state authorizing the display of programs. And, the same smartTV:TA taking on the configured state enacts a prohibition rule preventing the display of programs. How to consider the adaptability of rule enactment is another question to address.

Our contributions are as follows: (i) abstraction of TAs into assets offering insights into how, why, when, and where TAs are used, (ii) association of TAs with ODRL policies allowing a fine-grained definition and control, (iii) integration of ODRL policies into TAs discovery and composition, which means both better satisfaction of IoT applications’ requirements and better awareness of TAs’ capabilities and limitations, (iv) analysis of ODRL policies to confirm the participation of TAs in compositions, and (v) demonstration of the technical doability of ODRL-based discovery and composition of TAs. The rest of this paper is organized as follows. Section 2 defines concepts and presents related works. Section 3 details how ODRL impacts TAs discovery and composition. This impact is technically demonstrated in Section 4. Section 5 includes the conclusion and future work.

## 2 BACKGROUND

This section presents ODRL, TAs, and some works.

### 2.1 ODRL

ODRL is an example of rights expression language that provides a flexible and interoperable information model, vocabulary, and encoding mechanisms to represent statements about the usage of assets. An asset is an identifiable resource or a collection of resources such as data/information, content/media, applications, and services. ODRL relies on policies to represent permitted and prohibited actions over a certain asset as well as required obligations that stakeholders must meet over this asset as well (De Vos et al., 2019; W3C, 2018). ODRL information model’s key constructs are:

- Policy could include one to many permission, prohibition, or duty rules. First, permission allows an action over an asset if all constraints are satisfied and if all duties are fulfilled. Second, prohibition disallows an action over an asset if all constraints are satisfied. Finally, duty forcibly exercises an action over an asset or not. It is fulfilled when all constraints are satisfied and have been exercised.
- Party is an entity or a collection of entities that could correspond to a person, group of persons, organization, or agent. A party can fulfill different roles including assigner, assignee, informed party, consented party, consenting party, compensated party, and tracked party.
- Constraint is used either to refine components like action, party, or collection of assets or to declare conditions applicable to a rule. Constraints could be combined using logical operators.

In JSON-LD Listing 1, the assigner (line 7) in charge of an asset, *movie1* (line 6), refers to an agreement policy (line 4) consisting of 2 rules. The permission rule (line 5) is concerned with the *display* action (line 9) that an assignee, *smart TV* (line 8), executes subject to satisfying the constraint that is not enabling the permission rule for more than 4 hours (lines 11-15). And, the prohibition rule (line 17) is concerned with the *digitize* action (line 21) and is associated with the same assignee (line 20). Should the assignee execute the *digitize* action, then the remedy (line 22) would be to apply the *anonymize* action (line 23) to the asset *movie1* (line 24).

Listing 1: ODRL specification of movie.

```

1 {
2   "@context": "http://www.w3.org/ns/odrl.jsonld",
3   "uid": "http://example.com/policy:01",
4   "@type": "Agreement",
5   "permission": [{
6     "target": "http://example.com/asset:movie1/",
7     "assigner": "http://example.com/MovieParty:org:abc",
8     "assignee": "http://example.com/smart-TV/",
9     "action": "display",
10    "constraint": [{
11      "leftOperand": "meteredTime",
12      "operator": "iteq",
13      "rightOperand": {"@value": "P4H", "@type": "xsd:duration"},
14      "unit": "https://www.wikidata.org/wiki/Q25235"
15    }],
16  },
17  "prohibition": [{
18    "target": "http://example.com/asset:movie1/",
19    "assigner": "http://example.com/MovieParty:org:abc",
20    "assignee": "http://example.com/smart-TV/",
21    "action": "digitize",
22    "remedy": [{
23      "action": "anonymize",
24      "target": "http://example.com/asset:movie1/"}]
25  }

```

## 2.2 Thing Artifact

We briefly define the concepts of functionality, life cycle and interaction flow (Maamar et al., 2023).

**Definition 1.** A TA is defined by the tuple  $\langle f, a, D, lc, IF \rangle$  where  $f$  is either a primitive functionality  $f_p$  or a composite functionality  $f_c$  that the TA provisions;  $a$  is an action that a third party exercises on a TA's functionality  $f$ ;  $D$  is a set of input data  $\{d_i^{in}\}$  and output data  $\{d_j^{out}\}$  related to the TA;  $lc$  is the TA's lifecycle represented as a state diagram (Definition 2); and  $IF$  is a set of interaction flows  $\{if_i\}$  that are put together on-the-fly based on messages that the TA sends to peers and receives from peers (Definition 3).

**Definition 2.** A TA's lifecycle  $lc$  is defined by the couple  $\langle S, T \rangle$ , i.e.,  $lc = s_i \xrightarrow{trans_i} s_{i+1} \xrightarrow{trans_{i+1}} s_{i+2} \dots s_{j-1} \xrightarrow{trans_{j-1}} s_j$ ;  $S$  is a set of couples  $\{(s_i, \{P_{ij}\})\}$  where  $s_i$  is a particular state and  $\{P_{ij}\}$  is a set of ODRL policies associated with this state; and,  $T$  is a set of transitions  $\{trans_i\}$ .

**Definition 3.** A TA's interaction flow  $if_i$  is a set of messages  $\{m_{ij}\}$  that the TA uses to interact with peers. A message  $m$  is defined by the tuple  $\langle id, type, from/state, to/state, cnt \rangle$  where  $id$  is a message's identifier;  $type$  is a pre-defined message;  $from/state$  is the sending state in the TA's lifecycle;  $to/state$  is the receiving state in the lifecycle of a peer to the TA; and  $cnt$  is a content of data conveyed from the TA to a peer where  $cnt \subseteq D$ .

For more details about TAs integration into IoT applications, readers are invited to consult (Maamar et al., 2023).

## 2.3 Related Work

Our current work is at the crossroads of 3 topics, ODRL (Section 2.1), DA, and IoT discovery and composition techniques. To the best of our knowledge, such techniques for TAs do not exist.

**Data Artifacts in Brief.** In (Kumaran et al., 2008), Kumaran et al. propose guidelines that would assist IT practitioners identify the necessary DAs for their applications. In (Narendra et al., 2009), Narendra et al. model BPs using context-based artifacts and Web services. The authors abstract processes into models that are expressive (i.e., easy to grasp) for non-IT practitioners and could be based on DAs. In (Maamar et al., 2010), Maamar et al. discover and model DAs from business requirements. They use a bottom-up analysis to determine fine-grained data, which are

afterwards aggregated into clusters where each cluster becomes a potential DA. Next, they derive the operations that act upon the discovered data clusters. DAs are obtained after grouping the data and operation clusters together. Finally, in (Popova et al., 2015), Popova et al. acknowledge the role of DAs in modeling BPs and propose a set of methods to discover DAs' lifecycles. The methods are implemented as software plug-ins for ProM [www.promtools.org](http://www.promtools.org), a generic open-source framework for supporting process mining. Fig. 1 represents a purchase order based on DAs.

**Thing Discovery Techniques in Brief.** In (Sunthonlap et al., 2018), Sunthonlap et al. present Social-Aware aNd Distributed (SAND) scheme for device discovery in an IoT ecosystem. Among the criteria for ranking potential devices, the authors cite social relationship diversity with reference to a glucose-level monitor that sends requests to find a glucose analyzer with whom it could collaborate to evaluate a given patient's glucose level. The work of Sunthonlap et al. is in line with the trend of social IoT that blends social computing with IoT (Atzori et al., 2012; Maamar et al., 2020b). In (Barnaghi and Sheth, 2016), Barnaghi and Sheth mention that IoT applications cannot use existing data and service search, discovery, and access methods and solutions because of the number of resources and heterogeneity and complexity of data. This no use is also backed by Khalil et al. (Khalil et al., 2020). To this end, Barnaghi and Sheth put forward some requirements and challenges for developing a robust and comprehensive IoT search solution. Requirements include quality, latency, trust, availability, reliability, and continuity that should impact efficient access and use of IoT data and services. And, challenges result from today's IoT ecosystems that host billions of dynamic things that make existing search, discovery, and access techniques inappropriate for IoT data and services. In (Pablo Calcina et al., 2016), Pablo Calcina et al. acknowledge that finding useful devices to perform tasks is a major challenge for IoT. They examine the impact of network topologies, centralized, decentralized, and hierarchical, on device discovery in terms of discovery time, server-side infrastructure, reliability, and network traffic. To address security, privacy, and semantic issues when discovering things, W3C recently recommends Web of Things (WoT) Discovery standard (W3C, 2023) to allow customers to get a thing's Thing Description (TD) that could be registered in a server, for example. The discovery process in-

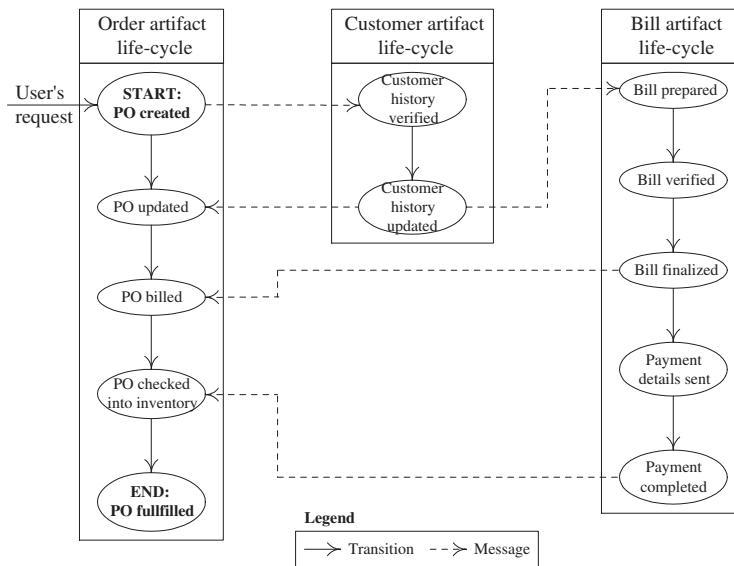


Figure 1: Representation of a DA-based purchase-order BP (Narendra et al., 2009).

cludes 2 phases. The introduction phase relies on existing mechanisms (e.g., URI and DNS-based) to request a thing's TD address (URL). And, the exploration phase uses this address to provide the TD after authenticating the requesting customer. Finally, an extensive analysis of discovery techniques in IoT is presented in (Khalil et al., 2020). These techniques are classified into content-, context-, location-, user-, semantic-, and resource-based. Despite their multiplicity and variety, constraints on things are overlooked during discovery. For instance, a content-based technique targets readings taken by a sensor. However, these readings only happen between 2am and 4am, while a future IoT application requires continuous readings making this sensor unfit for the application. Also, a location-based technique targets particular locations of users and not things to discover. Invoking a thing must happen from a specific location but this is overlooked in this technique.

#### Thing Composition Techniques in Brief.

In (Khaled and Helal, 2018), Khaled and Helal propose a programming framework to capture inter-thing relations into IoT applications. The framework ensures that, on top of things' services, logical and functional ties between services are not overlooked by introducing 3 primitives that are thing service, thing relation (e.g., control/controlled by, drive/driven by, and extend/extended by), and recipe. In (Krishna et al., 2019), Krishna et al. acknowledge the difficult job of composing things due to their het-

erogeneity and suggest a Web-based tool called IoT Composer. The tool assists users select, configure, and bind things together. In (Åkesson et al., 2019), Åkesson et al. compose services hosted on IoT devices using ComPOS (Composition language for Palcom Oblivious Services). Native services contain computation and interaction with the physical world, while composition services combine native services into applications, mediating and adapting messages between them. In (Ronny et al., 2021), Seiger et al. suggest HoloFlows to address the complexity of developing IoT applications for users and domain experts with limited technical background. Using HoloFlows, physical sensors and actuators are mapped onto virtual components to connect together through virtual wires. In (Maamar et al., 2020a), Maamar et al. examine thing composition in the context of thingsourcing. They argue that by analogy with people who participate in crowdsourcing, things could "act in the same way" leading to the formation and management of a crowd of things from which a group of selected things would be composed and assigned users' demands to complete. Finally, Noura and Gaedke present the Web of Things Description Language (WoTDL) ontology to describe key concepts associated with AI planning for automatic WoT composition (Noura and Gaedke, 2019). For Noura and Gaedke, WoT is the result of integrating IoT into the Web.

It is clear that the focus of the techniques above is on functionality, only, while other factors that en-

able/disable a functionality are overlooked. Permitting a functionality subject to payment and prohibiting a functionality subject to location mean better component discovery and composition. This is discussed in the next section where ODRL policies define the factors impacting TAs discovery and composition.

### 3 THING ARTIFACTS PROVISIONING

After presenting a case study, we dedicate the remaining sections to TAs discovery and composition.

#### 3.1 Case Study

Our case study is about a healthcare facility for elderly persons. Several studies confirm the benefits of IoT for these people for instance *IoT for seniors: Solutions and Use Cases* by Cogniteq<sup>1</sup> and *How can IoT help with elderly care?* by Telefónica Tech<sup>2</sup>. A first situation in the facility's smart living-room exposes 3 TAs, remoteControl:TA, smartTV:TA, and lightSwitch:TA, providing together a free-of-interruption watching experience to a group of persons. The remoteControl:TA is also used to control the living room's blinds upon the request of the smartTV:TA. Another situation exposes 2 extra TAs, smartWatch:TA and medicalDispenser:TA, synchronizing the automatic release of medicine pills to a person.

Besides the TAs' functionalities above like controlling lights, displaying movies, and dispensing medicines, invoking them to satisfy elderly persons' demands could be either permitted, prohibited, or even forced depending on for instance, time of the day, nature of medicine, and duration of a treatment. For instance, a medicalDispenser:TA is discovered because its daily power charging does not conflict with the periods of dispensing medicines. This charging is integrated into the medicalDispenser:TA's ODRL permissions. And, a smartTV:TA is composed because it responds to the unique demands of a particular remoteControl:TA integrated into this smartTV:TA's ODRL duties. We discuss how similar situations and their respective permissions, prohibitions, and duties impact TAs discovery and composition.

#### 3.2 Approach

Fig. 2 illustrates the approach to discover and compose TAs. It runs over 2 stages (*preparatory* and *development*), uses 5 modules (*creator*, *specifier*, *discoverer*, *checker*, and *composer*) and 2 repositories (*TAs* and *policies*), and, finally enables interactions across all modules and across modules and repositories. 2 stakeholders (*engineer* and *user*) participate in the approach as well.

In the *preparatory* stage, the engineer invokes (1) the *creator* to specify (2) TAs that reside in the ecosystem like smart living-room. The engineer could resort to some W3C standards like the WoT Thing Description (WoT-TD (W3C, 2023)) to specify TAs. However, this does not fall into the scope of the current work. Next, details about TAs' functionalities and lifecycles are stored in the *repository of TAs* (3). The content of this repository is made available for the *specifier* that with the assistance of the engineer (4) defines (4) and stores the necessary TAs' ODRL policies in the *repository of policies* (5) concluding the *preparatory* stage. Because TAs' interaction flows are generated at runtime, they are discussed in the *development* stage.

In the *development* stage, functionalities to select, lifecycles to specify, and interactions to initiate are IoT-application dependent aiming at satisfying particular users' demands such as controlling TV programs (6). First, based on an IoT application's requirements, the *discoverer* pulls details from both the *repository of TAs* and the *repository of policies* to discover and select the necessary TAs as per Section 3.2.1. This selection considers, on top of TAs' functionalities, the permissions, prohibitions, and obligations of how, when, where, and why to invoke these functionalities. Before the *checker* confirms the participation of the necessary TAs (7) in the IoT application as per Section 3.2.2, the *checker* analyzes how the interaction flows between the selected TAs' lifecycles would progress at run-time with respect to their respective ODRL policies. It could happen that a TA is not among those eligible to invoke a peer's functionality or that a peer's functionality has some invocation conditions that a TA cannot satisfy. As a result, either the TA or the peer is discarded from the under-development IoT application initiating the discovery of another TA. After the participation confirmation, i.e., all ODRL policies are satisfied, the *checker* notifies (8) the *composer* of the readiness of the necessary TAs to deploy on run-time platforms (9).

##### 3.2.1 Discovery

Existing discovery techniques of software components like Web services (Cheng et al., 2017) and hard-

<sup>1</sup><https://tinyurl.com/yae5dmv9>.

<sup>2</sup><https://tinyurl.com/yc7neuj7>.

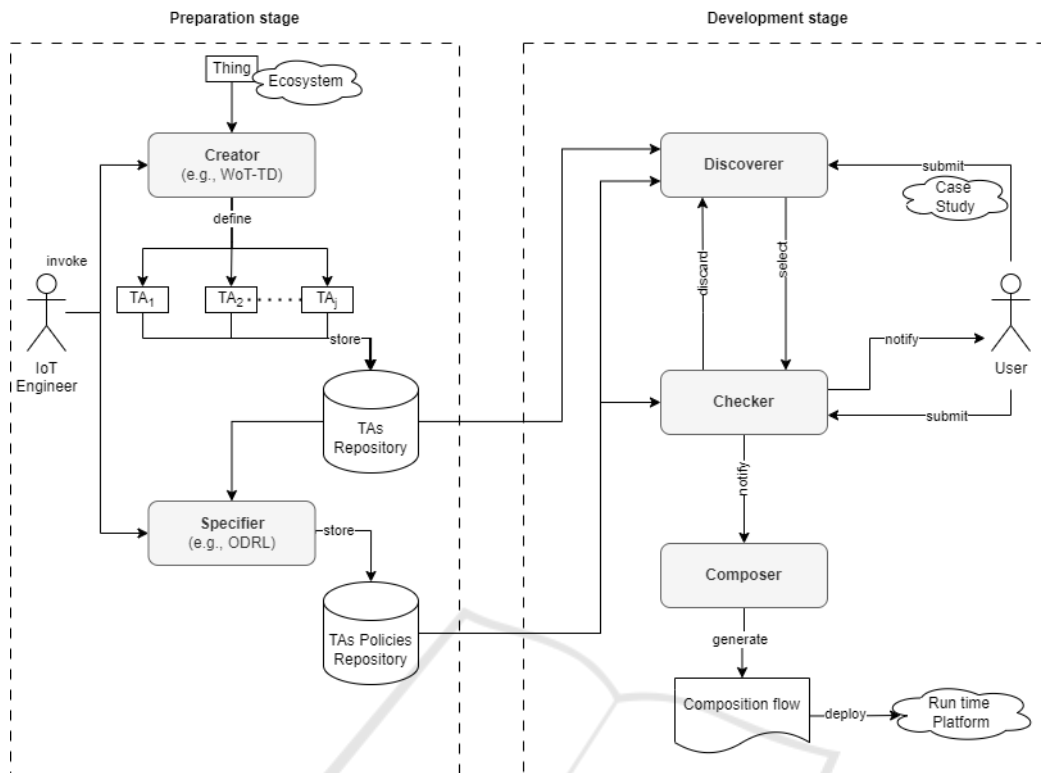


Figure 2: General representation of TAs discovery and composition approach.

ware components like IoT devices (Kamel et al., 2021) extensively depend on these components' functionalities<sup>3</sup> that, in fact, correspond to the capabilities to respond to users' demands. For instance, a Web service is discovered because it reports zip code-based weather forecasts and an IoT device is discovered because it measures humidity levels in a warehouse. To improve the outcome of discovery, some techniques reconcile semantic heterogeneities and/or tap into social relations between components. While discovery techniques safely assume a full access to the components' functionalities, they do not pay much attention to the conditions on both the functionalities and the actions invoking these functionalities. Although a discovered weather Web service is a "strong" candidate for integration into a road-traffic monitoring application, it will be discarded. The store action that invokes this Web service is conditioned by a specific format and duration. However, these two conditions do not meet the monitoring application's storage requirements that refer to a different format and duration. As a result, another Web service whose store action's conditions match the storage requirements will be selected after discovery. In our work, conditions on

<sup>3</sup>Criteria like quality-of-service and/or quality-of-thing could narrow down the discovery. However, this is not discussed further in this paper.

functionalities and actions are part of the permission, prohibition, and obligation rules in ODRL policies.

Back to Fig. 2 where the *discoverer* analyzes the ODRL policies stored in the *repository of policies* according to the under-development IoT application's requirements. These policies refer to TAs that the *discoverer* will recommend to the *composer*, through the *checker*, for inclusion in the IoT application. To illustrate this analysis, let us consider Listing 2 related to a smartTV:TA's functionality that is smartTV:volume:TA (line 7) with focus on how to amplify the volume using the *increase* action (line 10). In this Listing, a prohibition rule restricts to amplify the volume above a certain limit (lines 12-14) during night time (lines 15-20). Should an IoT application be flexible with any volume at any time of the day, then this smartTV:TA would be dropped from the list of candidate TAs for inclusion in this application. Night-time period restricts the volume level, which is not in line with the IoT application's requirements. As a result, the search for another TA continues.

Listing 2: ODRL specification of smartTV:volume:TA.

```

1 {
2   "@context": "http://www.w3.org/ns/odrl.jsonld",
3   "@type": "Set",
4   "uid": "http://example.com/Policy:123.5dab2b8bae0A",
5   "profile":
6     "http://example.com/odrl:profile:055",
7   "prohibition": [{

```

```

8  "target": "http://example.com/asset:smartTV:volume:TA"
9  "assigner": "http://example.com/org:321",
10 "action": {
11   "rdf:value": { "@id": "odrl:increase"},
12   "refinement": [
13     { "leftOperand": "count",
14       "operator": "gteq",
15       "rightOperand": "10"},
16     { "leftOperand": "dateTime",
17       "operator": "gteq",
18       "rightOperand": { "@value": "T19:00Z", "@type":
19         "xsd:time" }},
20     { "leftOperand": "dateTime",
21       "operator": "iteq",
22       "rightOperand": { "@value": "T08:00Z", "@type":
23         "xsd:time" }},
24     { "leftOperand": "context",
25       "operator": "eq",
26       "rightOperand": { "activated" }}]
27 }

```

### 3.2.2 Composition

Contrarily to what is discussed in Section 2.3 where the emphasis is on thing composition techniques like orchestration *versus* choreography applied to TAs, we discuss hereafter how the participation of a discovered TA in an IoT application is either approved or rejected. This would be based on 2 ODRL constructs: (i) policies where some are free of assignees and (ii) potential conditions on rules and/or actions invoking TAs' functionalities.

Assuming that an IoT application's composition schema is already set, e.g., which TAs perform first and what data TAs exchange, we exemplify the composition schema with 2 TAs and then, analyze their approval/rejection's impacts on completing this composition schema. Fig. 3 is a partial representation of the respective lifecycles of remoteControl:OnOff:TA and smartTV:OnOff:TA where inter-state messages (dashed lines) form an interaction flow between these lifecycles at run-time. We recall that TAs' lifecycles consist of 5 similar states, *prepared*, *activated*, *suspended*, *failed*, and *done*. However, the sub-states in the *activated* state capture the particularities of each TA where *used* and *unused* are sub-states for remoteControl:OnOff:TA for example, and *turned-off*, *turned-on*, and *configured* are sub-states for smartTV:OnOff:TA for example. Having these 2 TAs included in an IoT application for watching movies would be driven by the exchange of messages between their respective *activated* states. Specific messages are used as per Table 1.

To begin with, we suggest an ODRL policy for smartTV:OnOff:TA in Listing 3. Major highlights of this policy are as follows: agreement type (line 3), prohibition rule (line 6), list of assignees (lines 9-11), *turn-on* action to invoke the functionality (line 13), state-based conditions on the rule (lines 19-22), and location-based conditions on the action (lines 14-

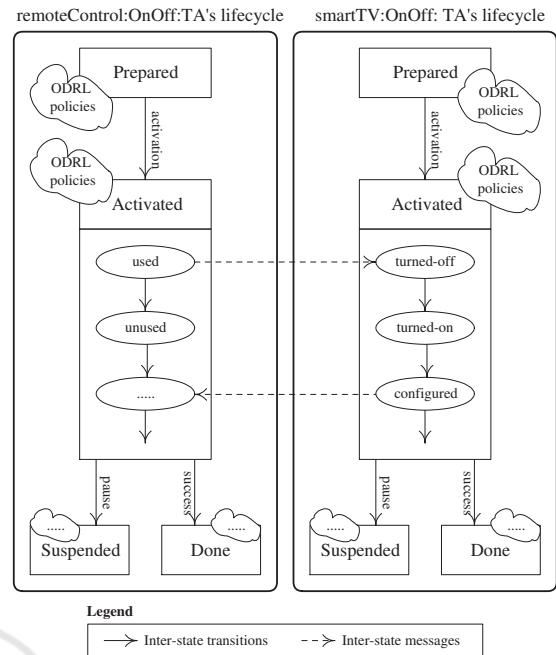


Figure 3: Representation of TAs engaged in interactions.

17). Let us assume a message from remoteControl:OnOff:TA's *activated:used* to smartTV:OnOff:TA's *activated:turned-off* requesting to execute the *turn-on* action. Upon message receipt, smartTV:OnOff:TA loads the ODRL policies linked to *activated:turned-off* state and proceeds as follows:

- If remoteControl:OnOff:TA is not in the list of assignees, then this TA will be rejected from the IoT application initiating the search for remoteControl:OnOff:TA' as per Section 3.2.1. Otherwise, checking rules and actions is initiated.
- Assuming that the state-based condition on the rule is satisfied, i.e., the current state is activated, checking the location-based condition on the action is next to ensure that invoking this action happens from a specific location. If the check turns out unsatisfied, i.e., remoteControl:OnOff:TA is in another location, then this TA will be rejected from the IoT application initiating the search for remoteControl:OnOff:TA' as per Section 3.2.1. Otherwise, remoteControl:OnOff:TA is approved for the composition.

Listing 3: ODRL specification of smartTV:OnOff:TA.

```

1 {
2   "@context": "http://www.w3.org/ns/odrl.jsonld",
3   "@type": "Agreement",
4   "uid": "http://example.com/policy:004",
5   "profile": "http://example.com/odrl:profile:05",
6   "prohibition": [
7     { "target": "http://example.com/smart-TV/smartTV:OnOff:TA",

```

```

8  "assigner": "http://example.com/smart-TVProvider:001",
9  "assignee": {
10   "@type": "PartyCollection",
11   "source": "http://example.com/RemoteControlList/
12     List1",
13   "action": [{
14     "rdf:value": {"@id": "odrl:turn-on"},
15     "refinement": {
16       "leftOperand": "spatial",
17       "operator": "eq",
18       "rightOperand": {"X"}}
19   "constraint": {
20     "leftOperand": "context",
21     "operator": "eq",
22     "rightOperand": {"activated"}},
23   "remedy": {
24     "action": "encrypted/scrambled transmission",
25     "target": "http://example.com/smart-TV/smartTV:
26       display:TA"}]}]}
27   }
28   "@type": "dc:Document",
29   "@id": "http://example.com/RemoteControl/
30     remoteControl:OnOff:TA",
31   "dc:title": "remoteControlsTA",
32   "odrl:partOf": "http://example.com/RemoteControlList/
33     List1"

```

## 4 IMPLEMENTATION AND EVALUATION

To demonstrate ODRL-based discovery and composition of TAs using the center for elderly persons, a system was developed, deployed, and evaluated on a Windows 11 desktop, 16GB RAM, 4 Cores processor, and 4 Cores Graphics card with 8GB VRAM (Fig. 4). We used *Visual Studio Code* (code.visualstudio.com, a free open-source lightweight code editor for JavaScript and TypeScript languages), the progressive JavaScript framework *vue* (vuejs.org, an open-source model–view–viewmodel front-end JavaScript framework for building user interfaces and single-page applications), *Node.js* (nodejs.org/en, an asynchronous event-driven JavaScript runtime for creating networks based on asynchronous operations), *Thingweb Node-WoT* (projects.eclipse.org/projects/iot.thingweb) package (an official reference implementation of the WoT), *Eclipse Mosquitto* (mosquitto.org, an open source message broker), and *ODRL validator and evaluator sandbox* (odrlapi.appspot.com) (an online Web application for verifying the status of ODRL policies).

In Fig. 4's left panel, the system offers Thing artifacts, Policies, and Discovery and composition flow options. First, through the Thing artifacts option that implements the *creator*, the engineer defines TAs using WoT-TD and stores their definitions in the *repository of TAs*. Regarding TAs' lifecycles and interaction flows, they are included in WoT-TD's *properties* attribute and WoT-TD's *actions* and *events* attributes, respectively. Second, through the Policies option that implements the *specifier*, the engineer de-

fines the TAs' policies to store in the *repository of policies*. For illustration, Listing 4 corresponds to the WoT-TD of smartTV:volume:TA while referencing the necessary constructs of policies in line 15. Details about each policy, e.g., line 18, are defined separately like in Listing 2. Finally, the Discovery and composition flow option implements the *discoverer*, *checker*, and *composer* according to the user's case study.

Listing 4: Example of a TA in WoT-TD including policies.

```

1  {
2  "@context": "https://www.w3.org/2019/wot/td/v1",
3  "@type": "Thing",
4  "title": "smartTV:volume:TA",
5  "id": "http://example.com/smartTV:volume:TA:8a0fdd027a62",
6  "securityDefinitions": {"basic_sc": {...}},
7  "security": ["basic_sc"],
8  "properties": {
9    "functionality": {
10     "@type": "Odr:prop",
11     "type": "string",
12     "value": "volume",...
13     "forms": [...]
14   },
15   "policies": {
16     "@type": "odrl:policies",
17     "type": "array",
18     "uid": ["http://example.com/Policy:123.5dab2b8bae0A",
19     ],
20     "forms": [...],
21     ...},...

```

To discover potential TAs, users are prompted to submit the functionalities they need like *toggle*, *volume*, and *control*. After a syntactical screening of the *repository of TAs* by the *discoverer*, some TAs are identified like smartTV:volume:TA, remoteControl:TA, and lightSwitch:TA (Fig. 5). Next, the user selects those TAs that she needs prior to setting certain requirements on some of these TAs like restricting smartTV:volume:TA to a maximum value (Fig. 5's right panel). The *checker* verifies the compliance of these requirements with TAs' policies like discussed in Section 3.2.1. No compliance would mean excluding the concerned TAs and selecting others.

After a successful check of all the TAs, the user with the assistance of the *composer* plugs the TAs together creating a composition flow (Fig. 6). Next, the *checker* verifies if the users' requirements satisfy the selected TAs' policies. Finally, the user assigns a name to the composition flow such as elderly person whose specification is shown in Fig. 6.

After completing the deployment, we evaluated the system in terms of flood attacks and response time. In the first evaluation, we considered HTTP-GET flood attacks *aka* distributed-denial-of-service. Some results are shown in Table 2. The response time remained stable even during the flooding, which meant that the server used the desktop's resources efficiently. We can also note that the server did not crash during the flooding, which also confirmed the efficient use of resources.



Table 1: Messages exchanged between TAs' lifecycles.

Type	Sender	Receiver	Description
<i>open</i>	$TA_i.ss_i$	$TA_j.ss_j$	Establishes a communication channel between the sender and receiver
<i>send</i>	$TA_i.ss_i$	$TA_j.ss_j$	Allows the sender to ask the receiver to invoke an action
<i>success</i>	$TA_j.ss_j$	$TA_i.ss_i$	Coupled with <i>send</i> ; confirms the acceptance of the action to invoke
<i>fail</i>	$TA_j.ss_j$	$TA_i.ss_i$	Coupled with <i>send</i> ; confirms the rejection of the action to invoke
<i>ping</i>	$TA_i.ss_i$	$TA_j.ss_j$	Checks periodically the liveness of the receiver
<i>ack</i>	$TA_j.ss_j$	$TA_i.ss_i$	Coupled with <i>ping</i> ; confirms the liveness of the receiver according to a specific delay
<i>close</i>	$TA_i.ss_i$	$TA_j.ss_j$	Coupled with <i>open</i> ; shut downs a communication channel

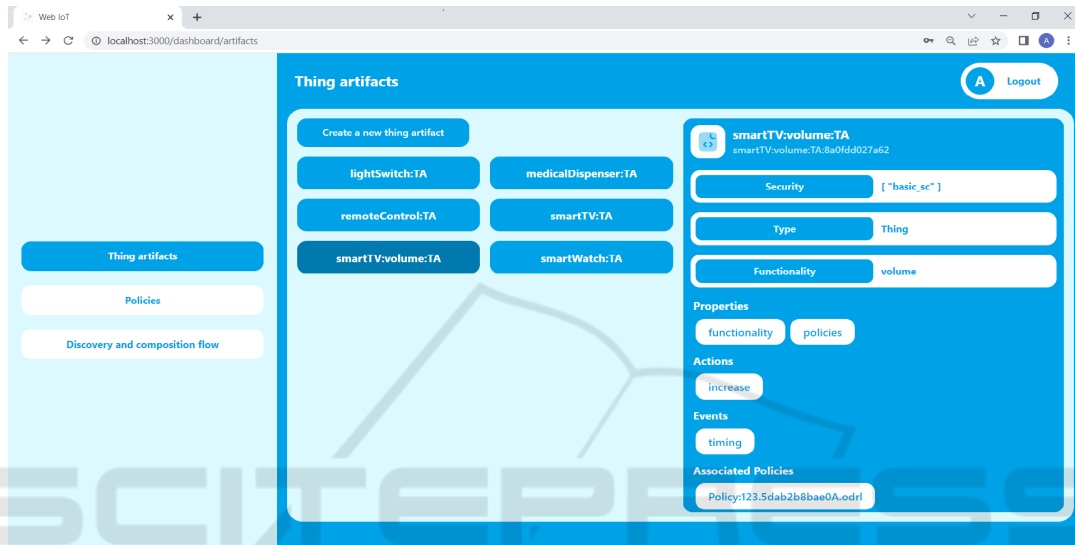


Figure 4: Main interface of the system.

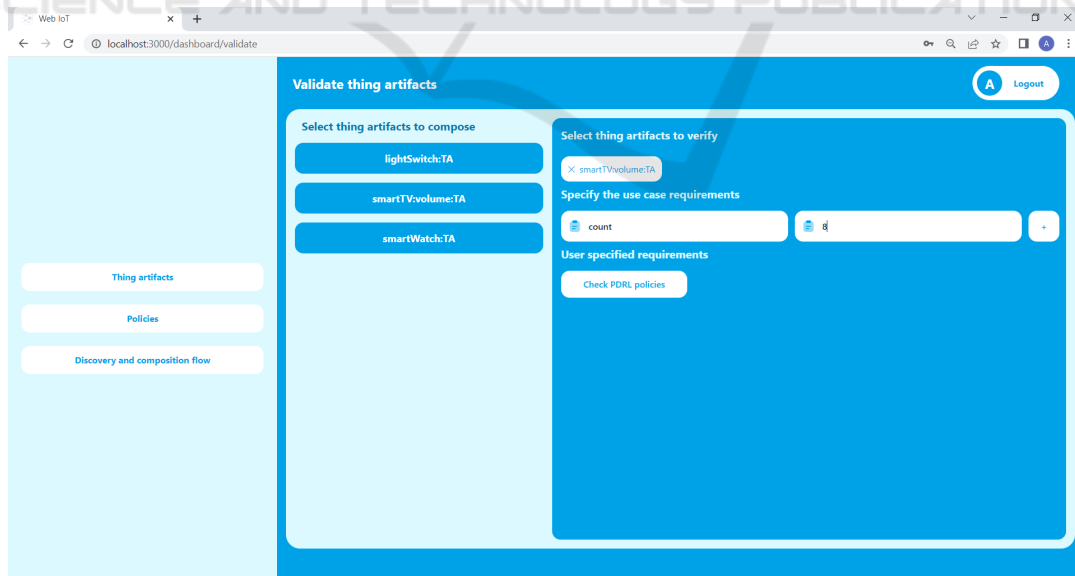


Figure 5: Screenshot of TAs discovery, selection, and requirement definition and check.

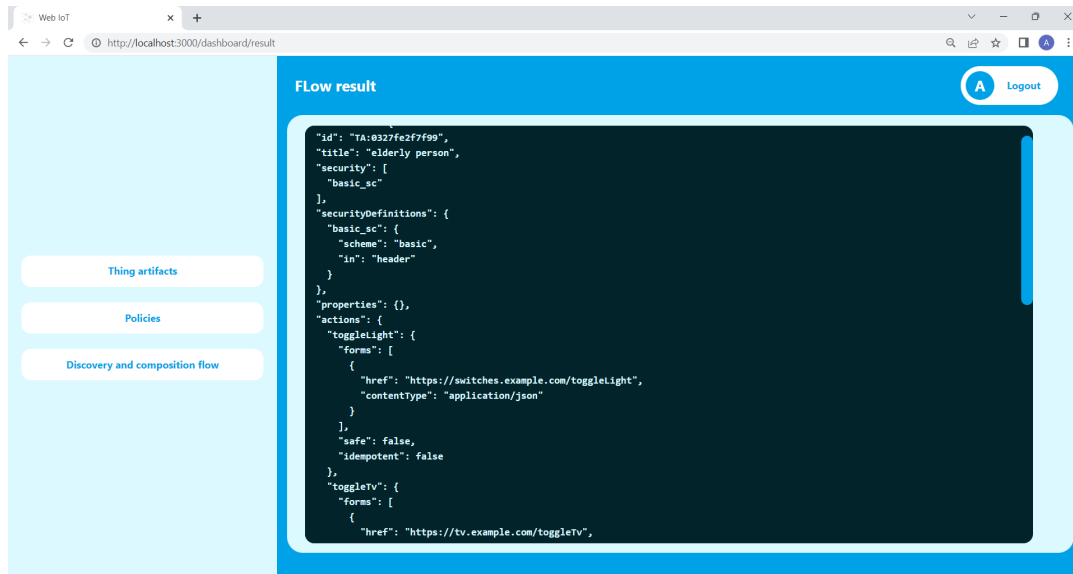


Figure 6: Screenshot of the specification of elderly person composition-flow.

Table 2: HTTP-GET flood test results.

Number of flood requests	Number of client requests	Average client response time (ms)	Server status
100	3	2.134	✓
1000	10	2.176	✓
10000	10	2.141	✓
100000	10	2.351	✓

✓ refers to server remaining functional

In the second evaluation, we considered the response time of each module of the system deployed on the desktop. The test consisted of a set of HTTP GET, POST, and DELETE requests that a client sends to a module via route endpoints. The results showed that the response time depends completely on the complexity of the operation that the route is executing. For instance, simple GET routes like policy and thing artifact home-routes had short response times (e.g.,  $\approx 0.5ms$ ) unlike the routes that perform complex operations like discovery (e.g.,  $\approx 0.9ms$ ) that uses *Node-WoT* to expose things and return their access links, which is a complex operation.

## 5 CONCLUSION

This paper presented a TA-based approach to design and develop IoT applications. Building upon the concept of data artifact, a TA is a chunk of unstructured information capturing 3 cross-cutting aspects referred to as functionality, lifecycle, and interaction flow. Along with defining these aspects, the approach integrated techniques to discover and compose TAs. A

major feature of these techniques is their compliance with organizations' operation rules specified as ODRL policies. Different illustrations about this compliance were included such as a monitoring TA that accepts invocations during a specific time-period while an operation rule approves invocations at any time of the day. This discrepancy would lead to dropping this TA from the list of candidate TAs to integrate in an IoT application. The technical doability of TAs discovery and composition was verified using an in-house system simulating some scenarios in a center for elderly persons. In term of future work, we would like to look into the impact of conflicting TAs's ODRL policies on the correctness of future IoT applications.

## REFERENCES

- Atzori, L., Iera, A., Morabito, G., and Nitti, M. (2012). The Social Internet of Things (SIoT) - When Social Networks Meet the Internet of Things: Concept, Architecture and Network Characterization. *Computer Networks*, 56(16).
- Barnaghi, P. M. and Sheth, A. P. (2016). On Searching

- the Internet of Things: Requirements and Challenges. *IEEE Intelligent Systems*, 31(6).
- Cano-Benito, J., Cimmino, A., and García-Castro, R. (2023). Injecting Data into ODRL privacy Policies Dynamically with RDF Mappings. In *Companion Proceedings of the ACM Web Conference 2023 (WWW'2023)*, Austin, TX, USA.
- Cheng, B., Shuai Zhao, S., Changbao Li, C., and Junliang Chen, J. (2017). A Web Services Discovery Approach Based on Mining Underlying Interface Semantics. *IEEE Transactions on Knowledge and Data Engineering*, 29(5).
- De Vos, M., Kirrane, S., Padget, J., and Satoh, K. (2019). ODRL Policy Modelling and Compliance Checking. In *Proceedings of RuleML+RR'2019*, Bolzano, Italy.
- Kamel, M., Yan, Y., Ligeti, P., and Reich, C. (2021). Attred: Attribute Based Resource Discovery for IoT. *Sensors*, 21(14).
- Khaled, A. and Helal, S. (2018). A Framework for Inter-Thing Relationships for Programming the Social IoT. In *4th IEEE World Forum on Internet of Things, WF-IoT2018*, Singapore.
- Khalil, K., Elgazzar, K., Seliem, M., and Bayoumi, M. (2020). Resource Discovery Techniques in the Internet of Things: A Review. *Internet Things*, 12.
- Krishna, A., Le Pallec, M., Mateescu, R., Noirie, L., and Salaün, G. (2019). IoT Composer: Composition and Deployment of IoT Applications. In *Companion Proceedings of ICSE'2019*, Montreal, QC, Canada.
- Kumaran, S., Liu, R., and Wu, F. Y. (2008). On the Duality of Information-Centric and Activity-Centric Models of Business Processes. In *Proceedings of the 20th International Conference on Advanced Information Systems Engineering (CAiSE'2008)*, Montpellier, France.
- Maamar, Z., Badr, Y., and Narendra, N. (2010). Business Artifacts Discovery and Modeling. In *Proceedings of ICSSOC'2010*, San Francisco, California, USA.
- Maamar, Z., Boukadi, K., Koné, B., Benslimane, D., and Elnaffar, S. (2020a). Thingsourcing to Enable IoT Collaboration. In *Proceedings of WETICE'2020*, France (online).
- Maamar, Z., Faci, N., Al-Khafajiy, M., and Murtada, D. (2023). Thing Artifact-based Design of IoT Ecosystems. *Service Oriented Computing and Applications*. (forthcoming).
- Maamar, Z., Faci, N., Kajan, E., Purkovic, S., and Ugljanin, E. (2020b). Process-of-Things: Weaving Film Industry's Practices into the Internet-of-Things. *Internet Things*, 11.
- Narendra, N., Badr, Y., Thiran, P., and Maamar, Z. (2009). Towards a Unified Approach for Business Process Modeling Using Context-Based Artifacts and Web Services. In *Proceedings of SCC'2009*, Bangalore, India.
- Nigam, A. and Caswell, N. (2003). Business Artifacts: An Approach to Operational Specification. *IBM Systems Journal*, 42(3).
- Noura, M. and Gaedke, M. (2019). WoTDL: Web of Things Description Language for Automatic Composition. In *Proceedings of WI'2019*, Thessaloniki, Greece.
- Pablo Calcina, C., Laisa Caroline Costa, D., Marcelo Knorich, Z., and Flávio Soares Corrêa, d. (2016). Device Discovery Strategies for the IoT. In *Proceedings of ISCE'2016*, Sao Paulo, Brazil.
- Popova, V., Fahland, D., and Dumas, M. (2015). Artifact Lifecycle Discovery. *International Journal of Cooperative Information Systems*, 24(1).
- Åkesson, A., Hedin, G., Nordahl, M., and Magnusson, B. (2019). ComPOS: Composing Oblivious Services. In *Proceedings of PerIoT'2019*, Kyoto, Japan.
- Ronny, S., Romina, K., Mandy, K., and Uwe, A. (2021). HoloFlows: Modelling of Processes for the Internet of Things in Mixed Reality. *Software System Model*, 20(5).
- Sunthonlap, J., Nguyen, P., Wang, H., Pourhomayoun, M., Zhu, Y., and Ye, Z. (2018). SAND: A Social-Aware and Distributed Scheme for Device Discovery in the Internet of Things. In *Proceedings of ICNC'2018*, Maui, HI, USA.
- W3C (2018). ODRL Information Model 2.2. <https://www.w3.org/TR/2018/REC-odrl-model-20180215/>. (Visited January 2021).
- W3C (2023). Web of Things (WoT) Thing Description 1.1. [www.w3.org/TR/wot-thing-description](http://www.w3.org/TR/wot-thing-description). (Visited December 2023).