An Ontological Framework for Opportunistic Composition of IoT Systems

Vatsala Nundloll, Yehia Elkhatib, Abdessalam Elhabbash, Gordon S. Blair
Distributed Systems Lab, School of Computing and Communications, Lancaster University, UK
Email: y.elkhatib@lancaster.ac.uk

***Abstract*—As the number of connected devices rapidly increases, largely thanks to uptake of IoT technologies, there is significant stimulus to enable opportunistic interactions between different systems that encounter each other at run time. However, this is complicated by diversity in IoT technologies and implementation details that are not known in advance. To achieve such unplanned interactions, we use the concept of a holon to represent a system's services and requirements at a high level. A holon is a self-describing system that appears as a whole when viewed from above whilst potentially comprising multiple sub-systems when viewed from below. In order to realise this world view and facilitate opportunistic system interactions, we propose the idea of using ontologies to define and program a holon. Ontologies offer the ability to classify the concepts of a domain, and use this formalised knowledge to infer new knowledge through reasoning. In this paper, we design a holon ontology and associated code generation tools. We also explore a case study of how programming holons using this approach can aid an IoT system to self-describe and reason about other systems it encounters. As such, developers can develop system composition logic at a high-level without any preconceived notions about low-level implementation details.**

1. Introduction

Facilitated by decreasing device costs, increasing connectivity reach and capacity, and a wide range of emerging applications (in business, health, security, etc.), the number of IoT devices has been rising increasingly and incessantly. A relatively early estimate [1] of the number of connected devices was to rise from 1 billion in 2009 to more than 26 billion by 2020. More recent figures [2] indicate that we already passed the figure of 26 billion connected devices in 2019. Further execution units, such as fog devices [3], are also being deployed to provide support roles closer to the end users [4].

This huge swarm of devices is made up of different systems (*e.g.* smart vehicles, smart cities, digital health, etc.) that are of varying technologies and operators. They co-exist in shared environments where the need for interaction between them is sometimes unplanned. Such opportunistic composition is needed to build more complex applications, *i.e.* ones that cross the boundaries of any single system, making future IoT applications more adaptable and, as such, more efficient and scalable.

However, developments in IoT systems have largely been *intra-system*, *i.e.* focusing on machine to machine communication (M2M) within the confines of a single IoT system. Little work has been done on enabling opportunistic *inter-system* interaction. An example of this is composing a system out of autonomous sub-systems, one managing a smart vehicle and another a smart house that had no a priori plan of interaction.

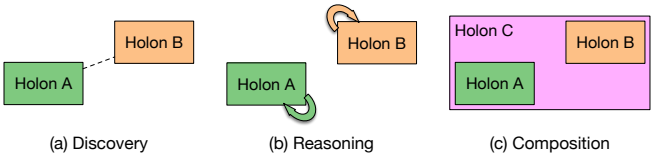

Figure 1. An overview of how holons opportunistically compose to form a more complex holon.

For such opportunistic interactions to take place, systems need to be managed at a high level in order to facilitate abstract system intentions. We previously introduced the concept of a *holon* [5], [6]: a self-describing system that appears as a whole when viewed from above (*i.e.* by the developer) whilst potentially comprising multiple sub-systems when viewed from below (*i.e.* by the runtime). This reduces the level complexity that a developer has to deal with, allowing them to focus on a system as a first-class programmatic abstraction that adapts to its operating environment without necessarily being impeded by the low-level internal details of the system and its components. In this manner, the holon concept forms the basis of an ecology where systems are able to opportunistically interact based on context and compose to create more complex holons arising spontaneously in a bottom-up manner (Fig. 1).

In this paper, we formalise this concept by presenting an ontology for developing holons, which in turn facilitates inter-IoT system composition through the 3 stages depicted in Fig. 1: discovery, reasoning, and finally composition. The ontology is defined in order to represent all system entities and use them to form holons, and to use such holons to compose higher level systems of systems, *i.e. super-holons*. The contributions of this paper are as follows:

1) Design of a domain ontology to represent a holon, using recognised ontologies for adaptive interaction in heterogeneous environments;
2) Means to generate relevant UML and Java interfaces;
3) Detailed presentation of how our ontological framework would work in a proof of concept IoT application.

The remainder of the paper is structured as follows: Section 2 presents a background on IoT ontologies; Section 3 details our ontology and how to use it to define a holon; Section 4 describes how to program a holon; Section 5 presents a proof of concept of how to use our ontology to build a complex application for the care of disabled patients; Section 6 discusses the presented work and its implications; and Section 7 concludes.

2. Background on IoT Ontologies

This section introduces ontologies, followed by an overview of ontologies for IoT and similar systems.

2.1. What are ontologies?

An ontology is a formal descriptive notation given to concepts that constitute a particular domain. Ontologies offer a simple but powerful notion used to classify concepts of a domain in the form of a superclass-subclass model and also to define the relationships that exist among these different concepts. Ontologies can describe the knowledge found in a domain and act as a structural framework in classifying and interpreting information. In addition to making domain assumptions explicit, ontologies also permit analysis and potential reuse of this domain knowledge. Enabling classified information to be shared as a common vocabulary across people and applications, ontologies are consequently paving the way towards building a supportive infrastructure for information exchange and discovery. Examples of ontology-based solutions are the Semantic Desktop [7], and Semantic Security Web Services (SSWS) [8].

2.2. Relevant ontologies

In search for an appropriate ontology to design a holon, we have considered a number of existing sensor and observation ontologies[1], namely OntoSensor, SWAMO, MMI Device, SensorML, A3ME, Sensei Observation and Measurement, and IoT-Lite. Observation ontologies have a similar set of concepts such as entity/feature of interest, measurements, and a context where these measurements apply. In the following, we give a brief overview of these ontologies.

OntoSensor builds a knowledge base of sensor data, their capabilities and measurements. However, it has not been updated since 2008 and its concepts and properties are quite disorganized, rendering the ontology unsuitable for use in other applications. SWAMO focuses on a sensor domain, and the processes to control sensors. Although it is actively maintained, few concepts look mixed, for example *acceleration* seems to be part of a *position* class. MMI Device is designed to model oceanographic devices, their measurements and capabilities, but does not seem to be a complete ontology. SensorML serves as a starting ontology for the MMI Device, but is no longer maintained. Sensei Observation and Measurement annotates sensor observation data, but it is not actively maintained either, and the properties do not seem to be clearly defined. The A3ME ontology [9] has been found suitable for the holon design due to its simple classification for self-description and discovery of devices and their capabilities in heterogeneous networks including resource-constrained sensor nodes. Another ontology called CoDAMOS [10] seems suitable to capture the requirements of a holon. The ontology is actively maintained, is simple and extensible.

Another interesting class of ontologies consists of ontologies created specifically for IoT systems. A representative example of this class is IoT-Lite, a lightweight ontology to represent IoT concepts, properties and services. It is an instantiation of the Semantic Sensor Network (SSN) [11] ontology, which describes different concepts of an IoT network and the relationships therein. These ontologies are very specific to IoT deployments whilst holons present a much wider notion that can include IoT and other system types. As such, we did not use these ontologies, but instead used A3ME and CoDAMOS as starting points as they better suit our holon needs.

3. Defining Holons

A holon is a unitary first-class programmatic entity that can be used to model any distributed system. This simple concept enables a developer to specify, manipulate, and reason about systems in a programmatic manner. It also enables the construction of new holons – *i.e.* systems of systems – through holon composition.

We now describe how to specify a holon from a semantic point of view, and then we outline how to use the CoDAMOS and A3ME ontologies in the design of a holon.

3.1. Specifications of a Holon

Below we specify a holon through describing its fundamental services and properties.

HOLON: ***has parent*** another holon; **has children** some other holon; **supports** service(s).

Membership properties of Holon: *has Children*, *is Member*. **Physical Properties of Holon:** *power level (infinite/finite)*; *location*; *mobile*; *OS*; root *access*.

SERVICE: requires properties; **desires** properties; **guarantees** properties; **may provide** properties; **has an / exposes** API (public URI).

PROPERTY: *At Least Once* **is Better Than** *Best Effort*; *At Most Once* **is Better Than** *Best Effort*; *Exactly Once* **is Better Than** *At Least Once*; *Exactly Once* **is Better Than** *At Most Once*. *At Least Once* **is Exactly** *At Most Once* (this constraint is subjective).

The subsequent sections show how the CoDAMOS and A3ME ontologies have been used to model the holon.

1. https://www.w3.org/2005/Incubator/ssn/wiki/Review_of_Sensor_and_Observations_Ontologies

3.2. Holon design using CoDAMOS

This section shows the different concepts of the CoDAMOS ontology, and highlights the different ontology entries that have been extended in order to accommodate the definition of a holon. The CoDAMOS ontology is divided under four basic concepts: **User**, **Environment**, **Platform** and **Service**. The **User** concept denotes a profile, a role, a mood; can make use of an I/O device; and defines a task. The task executes an activity, and uses a service. The **Environment** concept defines a location, a time period, and an environmental condition. The location can be relative/absolute, and can be an address. Examples of environmental conditions are temperature, pressure, humidity, lighting and noise. The **Platform** concept provides a service that is used by the **User**, and is related to the **Environment** concept through the ***hasEnvironment*** property. The **Service** concept has a service profile, a service model, and a service grounding. A **Software** concept links to **Service** through a property called ***providesService***, whilst a **Task** concept makes use of that **Service**. Other resources such as memory, network, power or storage can also be modelled, and the **Software** concept can be differentiated as middleware, OS, rendering engine or virtual machine.

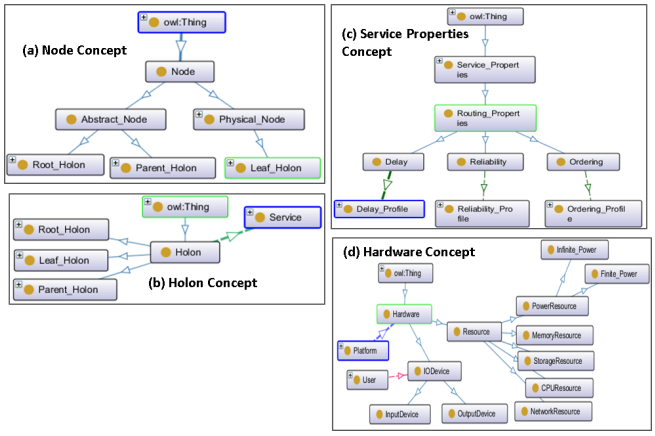

Figure 2. Holon Concepts

Fig. 2 demonstrates how the node and holon concepts have been defined in the ontology. Fig. 2(a) shows the **Node** concept, which is categorized as an abstract or a physical node. Fig. 2(b) depicts the **Holon** concept, classifiable as a `Root_Holon`, which has no parent but has a number of child holons; a `Parent_Holon` that can also be a root or have a number of children; and a `Leaf_Holon` with no children and cannot be a root. As such, root and parent holons are classified as abstract nodes as they can be further defined by a child node. A leaf holon is classified as a physical node as it does not have children.

Like any system, a holon has its own characteristics, among which are its routing properties. These are classified as **Service Properties**, as shown Fig. 2(c). This *Routing_Properties* concept is further been classified under 3 concepts: *Delay*, *Reliability* and *Ordering*, each of which has a profile namely *Delay_Profile*, *Reliability_Profile* and *Ordering_Profile* respectively. The above profiles are shown in Fig. 3(c). The Delay_Profile concept is defined as *Indefinite* or *Finite*. The *Finite* concept has been formulated with a relationship like *hasTime some Time*, and imposes some restrictions such as it has a minimum time limit of 1 Time unit and a maximum time limit of 4 Time units. This concept is displayed in Fig. 3(d). The Reliability_Profile, displayed in Fig. 3(e), is defined as At_Least_Once, At_Most_Once, Best_Effort and Exactly_Once. The concept At_Least_Once means that it is better than a *Best_Effort* profile, and can either be equivalent to an *At_Most_Once* effort depending on the requirements of a system. The concept *At_Most_Once* is better than a *Best_Effort* profile while the concept *Exactly_Once* is better than *At_Least_Once* and *At_Most_Once*.

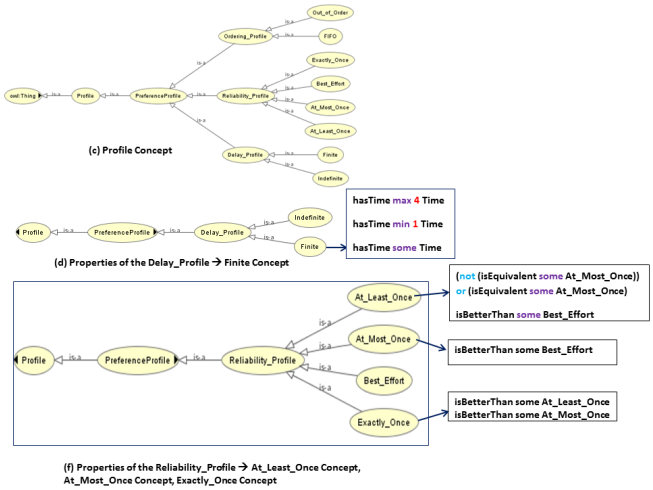

Figure 3. Profile Concepts

Finally, Fig. 2(d) displays how we can model the **Hardware** concept for a leaf holon, given that a leaf holon represents a physical node with various resources.

3.3. Holon design using A3ME

Given the similarities between the A3ME and CoDAMOS ontologies, and also given that A3ME details certain concepts better compared to CoDAMOS, we decided to further augment the CoDAMOS ontology using the A3ME ontology. The aim is to enhance the descriptions of certain concepts. Fig. 4 describes how the latter concepts have been enhanced. In Section 3.2, we mentioned that the leaf holon is defined as a physical node as it does not have any children. Fig. 4(a) shows that the ***Physical_Node*** concept from CoDAMOS can be extended using the ***Device*** concept from the A3ME ontology. Enabling the ***Physical_Node*** concept to become an equivalent class with the ***Device*** concept implies that this concept inherits all the properties of the ***Device*** concept, as shown below.

providesSoftware some OperatingSystem
providesHardware some NetworkResource
providesHardware some MemoryResource
providesHardware some PowerResource
providesHardware some StorageResource
providesHardware some CPUResource
providesHardware some IODevice
hasLocation some AbsoluteLocation

Similarly, the ***Storage*** concept from CoDAMOS can be extended using the equivalent ***Storage*** concept from A3ME, as shown in Fig. 4(b), thus inheriting concepts like ***Flash***, ***HD***, ***ROM*** and ***RAM***. The ***Power_Resource*** concept from CoDAMOS – which is divided only into Infinite and Finite Power concepts – has also been extended through the A3ME ***Energy*** concept into subclasses like ***Renewable***, ***Not_Limited***, ***Other_Energy***, ***Passive*** and ***Battery*** (Fig. 4(c)). The API_Public concept is extended into ***Global_Address***, ***Local_Address***, ***Other_Address*** from the ***Address*** concept in the A3ME ontology (Fig. 4(d)). Likewise, the Service concept can be further described using the Service concept from A3ME, thus inheriting subclasses like ***Real_world service***, ***Hardware service***, ***Other service*** and ***Software service*** (Fig. 4(e)).

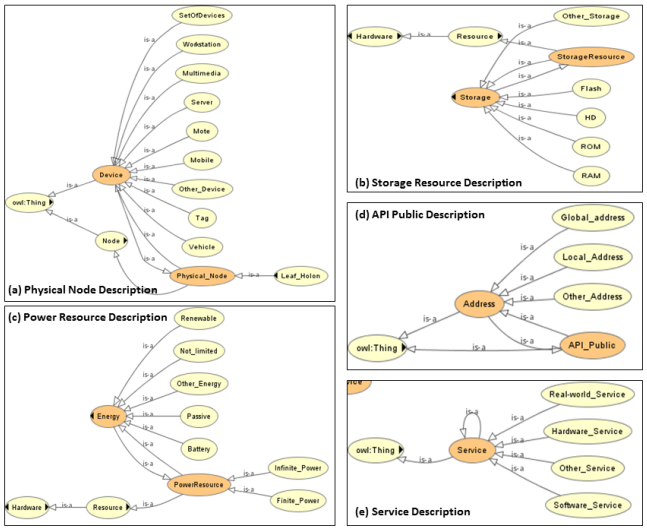

Figure 4. Enhanced Concepts

3.4. Holon ontology metrics

The result of developing the holonic ontology in the predefined manner is outlined through some metrics in Table 1, namely: the number of classes denoting the different created concepts; the number of object properties representing the relationships between different concepts; and the magnitude of data properties of numerical relationships the classes may hold. In summary, the resultant holonic ontology is considerably manageable especially considering the generality of its purpose.

TABLE 1. ONTOLOGY METRICS

Ontology Metric	*Count*
Class	66
Object property	28
Data property	18

4. Programming Holons

In order to create and manipulate a holon in a programmatic way, the concepts formalised by the above ontology need to be transformed into programming tools. In this section, we review two contrasting methods. First, in Section 4.1, we describe how we generated UML and Java interfaces from the ontology. Then, in Section 4.2, we describe a method of converting the ontology into a Domain Specific Language (DSL), which is intended for non-specialists to be able to program at a high-level in a specific application domain. We present both approaches as alternative routes to the task of holon implementation.

4.1. Converting ontologies into UML

The holon ontology has been designed using the widely used Protégé editor [12], which provides a few plugins to generate the UML and Java code that correspond to a given ontology. Whilst the A3ME ontology was found to work only with version 5.0 of Protégé, the CoDAMOS ontology was found to be compliant only with prior versions. Given that most of these plugins are compatible with older Protégé versions, the holon ontology using CoDAMOS only has been considered for this section. The used plugins are summarised in Table 2.

TABLE 2. PROTÉGÉ PLUGINS

Plugin	*Compatible Protégé version*
OWL2UML	4.0
Generate Protégé-OWL Java Code	3.4.7
Java Schema class	3.4.7
EMF Java Interfaces	3.4.7
Kazuki Java classes	3.4.7

All these plugins generate the corresponding Java code representing the ontology and can help programmers in their software development task. ***OWL2UML*** generates UML diagrams; ***Protégé-OWL Java Code*** generates Java Code; ***Java Schema class*** generates the Java schema class; ***EMF Java Interfaces*** generate Java interfaces; and ***Kazuki Java classes*** generate the schema of the ontology.

4.2. Converting ontologies into a DSL

On the other hand, DSLs are high-level programming languages that capture the behaviour specific to a certain application domain. DSLs are typically intended to allow non-programmers to create and edit software design specifications in an easy and clear way. As such, the domain

specialist can focus on *what* they want to achieve by expressing through the DSL, then the DSL code is parsed to a lower-level programming language to specify *how* to carry out the expressed logic. Recent application modelling DSLs include MixT [13] and CadaML [14].

We used OWL2GRA[2] [15] to convert the ontology into a DSL. OWL2GRA transforms an ontology to DSL grammar that can be used to capture and manage concepts in this domain. This model has three different modules:

1) Onto2OWL: creates an ontology file in OWL/XML format from a simple DSL language called OntoDSL [16], which provides a simple way to create ontologies. The resulting ontology can then be fed to software like Protégé to edit, manage, and further enhance the ontology.
2) OWL2DSL: generates a grammar specification (as well as an associated DSL parser) to describe domain elements.
3) DDesc2OWL: populates the ontology that was used to generate the grammar.

Of these modules, we only used OWL2DSL to convert our ontology into a DSL. Since we created our ontology using Protégé, we did not have a need to use Onto2OWL. We also did not need to use DDesc2OWL, however it might be useful for further experimentation.

5. Proof of Concept

We present and discuss a case study to illustrate how holons enable opportunistic system composition in IoT environments.

5.1. Background and challenges

DisabledCare is a system of IoT sub-systems that manages the living conditions of disabled patients in domestic environments. Each sub-system collects some data about the patient and their environment. For instance, one sub-system would collect measurements of ambient temperature and humidity. Another measures the levels of oxygen and carbon monoxide, etc. A further sub-system monitors the patient's body temperature and blood pressure. Another sub-system would track the patient's limited mobility patterns, and so on.

The DisableCare super-system (or system of systems) would analyse the data collected by all sub-systems, acting when necessary; *e.g.* to maintain certain environmental aspects or alert medical authorities. With the current advancements in digital health and the role of enabler the IoT plays in this field, it is no longer reasonable to assume such systems to be easily manageable by humans. In other words, the classical approach of deploying a system like DisabledCare by manually linking devices of distinct deployments is not practically feasible, certainly not at scale. Such classical approach requires IoT engineers to be familiar with (or spend significant time and effort to learn) the low-level details of the different devices and the services they each provide.

An alternative approach is to equip the IoT devices with capabilities that enable them to self-advertise their functionalities and self-discover neighbouring device functionalities. This consequently enables ad hoc interaction. The proposed ontology equips IoT systems with the means to harness such interaction towards automated composition. This case study shows how the proposed ontology can be utilised for this goal.

5.2. Architecture

Fig. 5 shows the architecture of DisabledCare, which consists of the following main components:

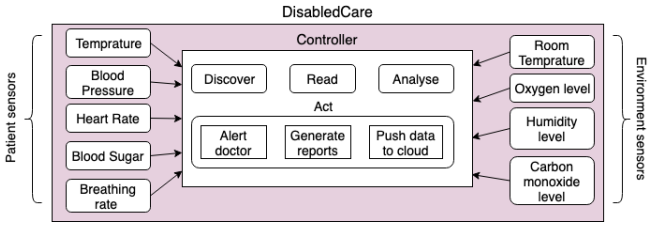

Figure 5. Architecture of the DisabledCare system

5.2.1. Patient sensors. These are a set of devices that read various vital signs from the patient's body. We show in Listing 1 part of the ontology of the *Temperature* device, which reads the temperate of the patient. As shown in the listing, this device has a service which can be invoked to obtain the temperature. The ontology shows also some of the context information in which the device operates correctly. For example, this device is designed to operate in an environment where the temperature falls between -10°C and 60°C. This information might be useful for applications using the *Temperature* device so that they identify when not to rely on the readings of this device.

Listing 1. Part of Temperate sensor's ontology

```
<owl:NamedIndividual rdf:about="#Temprature">
    <rdf:type rdf:resource="#Holon"/>
    <rdf:type rdf:resource="#Software"/>
    <Dionasys:providesService rdf:resource="#
        getTemprature"/>
</owl:NamedIndividual>
```

5.2.2. Environment sensors. These devices read data about the environment where the disabled patient exists. The device readings are useful to identify situations when the patient might need support or urgent medical attention. Listing 2 shows the ontology of the *Carbon Monoxide Reader* device, which provides the functionality `getLevel` that returns the carbon monoxide level of the location where the patient exists.

2. http://www4.di.uminho.pt/~gepl/OWL2GRA/

Listing 2. Part of Carbon Monixide devices's ontology

```
<owl:NamedIndividual rdf:about="#
    CarbonMonoxide">
    <rdf:type rdf:resource="#Holon"/>
    <rdf:type rdf:resource="#Software"/>
    <Dionasys:providesService rdf:resource="#
        getCarbonMonoxideLevel"/>
</owl:NamedIndividual>
```

5.2.3. Controller. This device hosts the application that reads the data from the patient and environment sensors, analyses them, and performs suitable actions. Listing 3 shows part of the ontology that describes the controller, focusing on its services (*e.g.* functionalities).

Listing 3. Part of the Controller's ontology

```
<owl:NamedIndividual rdf:about="#Controller">
    <rdf:type rdf:resource="#Holon"/>
    <rdf:type rdf:resource="#Software"/>
    <Dionasys:providesService rdf:resource="#
        alertDoctor"/>
    <Dionasys:providesService rdf:resource="#
        analyse"/>
    <Dionasys:providesService rdf:resource="#
        discover"/>
    <Dionasys:providesService rdf:resource="#
        generateReport"/>
    <Dionasys:providesService rdf:resource="#
        pushToCloud"/>
    <Dionasys:providesService rdf:resource="#
        read"/>
</owl:NamedIndividual>
```

5.3. Controller realisation

We now comment on the benefits of the proposed ontology in realising the functionality of the *Controller* device. We first describe the classical approach, then contrast it with the proposed approach using holons.

5.3.1. Classical approach. In this approach, IoT system engineers are expected to learn and understand the services, APIs, and properties of each of the patient and environment devices listed in Fig. 5. They then need to implement the `read` component of the *Controller* to manually connect to each of the services, which means that they need to design and develop the composition of the services provided by the devices. It is worth mentioning that IoT engineers will need to re-design or develop the composition in case any device needs to be replaced (*e.g.* in case of failure or an upgrade).

5.3.2. Envisioned approach. Here, IoT system engineers are just expected to implement the `discover` component of the *Controller* so that it discovers the sensors, reads their ontologies, identifies their service, and connects them to the `read` component. The *Controller* will then automatically add the service to its ontology as services it can provide indirectly by connecting to the devices. Listing 4 shows the services the *Controller* can provide after discovering the sensor services. In case of a device replacement, the `discover` component can detect the new device and connect it to the system in the fashion just described.

Listing 4. Part of the Controller's ontology after discovering devices

```
<owl:NamedIndividual rdf:about="#Controller">
    <rdf:type rdf:resource="#Holon"/>
    <rdf:type rdf:resource="#Software"/>
    <Dionasys:providesService rdf:resource="#
        alertDoctor"/>
    <Dionasys:providesService rdf:resource="#
        analyse"/>
    <Dionasys:providesService rdf:resource="#
        discover"/>
    <Dionasys:providesService rdf:resource="#
        generateReport"/>
    <Dionasys:providesService rdf:resource="#
        getBloodPressure"/>
    <Dionasys:providesService rdf:resource="#
        getBloodSugarLevel"/>
    <Dionasys:providesService rdf:resource="#
        getBreathingRate"/>
    <Dionasys:providesService rdf:resource="#
        getCarbonMonoxideLevel"/>
    <Dionasys:providesService rdf:resource="#
        getHeartRate"/>
    <Dionasys:providesService rdf:resource="#
        getHumidityLevel"/>
    <Dionasys:providesService rdf:resource="#
        getOxygenLevel"/>
    <Dionasys:providesService rdf:resource="#
        getRoomTemprature"/>
    <Dionasys:providesService rdf:resource="#
        getTemprature"/>
    <Dionasys:providesService rdf:resource="#
        pushToCloud"/>
    <Dionasys:providesService rdf:resource="#
        read"/>
</owl:NamedIndividual>
```

5.4. Reflection

The proposed ontological approach offers the ability for each sub-system of the DisabledCare system to richly describe itself. It also enables each sub-system to reason about its existence and how it fits into the bigger integrated system. In addition, the proposed approach provides the potential to relieve IoT engineers from the onus of developing and manually adapting IoT applications. Engineers will need to focus on the behaviour of the IoT application instead on being exhausted in learning the low level details of the used IoT devices.

6. Discussion

The holon notion sprang from the need to represent a system as an entity that expresses what its functionalities are, and also what it requires from other entities in order to provide added services. This is different from other approaches as we do not need to have an overarching ontology for all devices, *e.g.* [17], [18]. Holons are each self-described, facilitating the composition of systems into more complex systems of systems. We have demonstrated this using the digital health application presented in Section 5. Moreover, our approach allows complex applications to be built using pre-existing deployments, instead of trying to build and deploy a whole new system that spans various tasks (such as, in the case of digital health, environmental and patient-specific factors). This is encouraging for digital health IoT applications where multimodal sensing is of increasing significance [19], [20], but also for IoT applications in general as they share spaces in smart cities, homes, agricultural and manufacturing plants, etc.

Let us consider how overlays can be used in the context of IoT networks. If the interfaces of particular data sources from one IoT network can be discovered and used by other IoT systems, then this opens up the possibility of creating cross-IoT-deployment behaviour that was not possible before. For example, let us consider the case of three separate environmental IoT deployments: a weather monitoring network of devices with a GPRS connection, a soil moisture monitoring IoT and an ad hoc mobile IoT network of devices on sheep (*cf.* [21]). Questions that arise in this case are: What kind of overlay(s) can emerge from such networks? What should be the properties of the holon(s) representing each system? What type of composition, if any, should occur between a sheep tracking network overlay and a soil sensing network overlay, or between a sheep tracking network and a weather monitoring one? Through the composition of overlay structures, one can envisage a scenario where the sheep IoT can ferry data packets from the soil moisture IoT to the weather station as it comes into opportunistic contact with different weather stations endowed with data uplinks. The data collected and carried by the sheep IoTs can inherently be of use to particular weather station IoTs as they seek to correlate their own weather data with the sensor data from the soil moisture IoT. The use of semantically-enhanced data and ontologies will allow the weather station IoTs to make sense of the soil moisture data through an indirect source.

As powerful as it is, the holon notion also presents additional challenges. One such challenge is interoperability. For instance, to compose heterogeneous network overlays into a multi-overlay network [22], one needs to address issues such as connectivity and network interoperability. The fact that each overlay can potentially use different algorithms to maintain node membership need not be an obstacle but can be abstracted over. The challenge then involves how to dynamically manage members of different overlays and to designate a node acting on behalf of each overlay in forwarding data or implementing a type of communication pattern. In essence, these gateway nodes become the enablers of our 'system of overlays' at the network interoperability level [23], [24].

The design of these gateway nodes clearly requires a knowledge of systems involved, and also of potential systems encountered dynamically. It is inevitably a tedious job to design them, but we believe that the synthesis of these overlays becomes easier if they are modelled using high-level domain-specific languages enhanced using semantic technologies such as ontologies. All these queries lead us to investigate the enhancement of DSLs using ontologies such that we not only make our DSL more expressive, but also include some reasoning into our holon design.

7. Conclusion

Opportunistic composition of systems is required for unplanned and meaningful interaction between the myriad of systems including IoT deployments of all colours. The holon programming notion was proposed to facilitate opportunistic system composition through raising the level of abstraction when designing systems that could be made up of different individual sub-systems. However, an isolated programming notion is of little use if it is not easy to express programmatically and reason about, thus unfolding and managing composition opportunities.

This paper presented an OWL-defined ontological framework to support the holon programming abstraction by providing a conceptual structure that defines the domain of interest, with particular focus on IoT deployments, and describes the different concepts, relations, and rules therein. The ontology facilitates knowledge management in a holonic world, which in turn benefits interoperability, reuse and sharing. Such accessibility to information can be used to decide whether composition is possible between holons representing IoT and other types of systems. Moreover, the ontology also enables reasoning, such as analysing why and how specifically composition is possible (which is our immediate next step).

The paper also identified the benefits of merging such ontology with a DSL to define a holon. Existing ontologies can be used to initiate DSL development, which we have validated through the use of OWL2GRA to convert our ontology into a high-level language that models the holonic domain. What is of significance here is that such a conversion helps to stipulate reasoning, which is missing through manual DSL design methods. The ability to define classes, their properties and restrictions, and to handle incomplete knowledge through reasoning distinguishes OWL from class-based modelling languages like UML. We thus believe that our ontology can increase the expressiveness of the metamodeling language.

Whilst we have highlighted the role of using ontologies in enhancing a DSL, we have also pointed out how to represent the concepts of an ontology in form of a UML. A question arises here: whether a UML representation and a generated DSL are mutually exclusive, or if it is possible (and indeed beneficial) to combine them in order to get

the best out of both. This is an interesting future research direction.

We intend to expand on this work by enabling reasoning of the holon concepts using OWL-based reasoning engines, and incorporating this into a DSL model. The will enable SoS developers to describe their elementary systems, including functionalities and properties, using high-level ontological concepts. The elementary systems will then utilise their descriptions to dynamically construct SoSs at run-time. We plan to apply it to an IoT scenario and see how the OWL-enriched DSL can help in better defining holons and in composing different holons to suit the application needs. We also aim to use simulations (using IoTNetSim [25]) to investigate the suitability of our approach as IoT deployments grow in size and complexity.

Acknowledgments

This work was supported by the CHIST-ERA *Dionasys* project via EPSRC grant reference EP/M015734/1. The authors thank Laurent Réveillère and Etienne Rivière for their feedback on an earlier version of this work.

References

[1] Gartner, "Forecast: The internet of things, worldwide," http://www.gartner.com/document/2625419, 2013.

[2] Statista, "Internet of things (IoT) connected devices installed base worldwide from 2015 to 2025," https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/, 2019.

[3] Y. Elkhatib, B. F. Porter, H. B. Ribeiro, M. F. Zhani, J. Qadir, and E. Rivière, "On using micro-clouds to deliver the fog," *Internet Computing*, vol. 21, no. 2, pp. 8–15, Mar. 2017.

[4] Y. Elkhatib, "Building cloud applications for challenged networks," in *Embracing Global Computing in Emerging Economies*, ser. Communications in Computer and Information Science, R. Horne, Ed. Springer International Publishing, 2015, vol. 514, pp. 1–10.

[5] G. Coulson, G. S. Blair, Y. Elkhatib, and A. Mauthe, "The design of a generalised approach to the programming of systems of systems," in *Workshop on Autonomic and Opportunistic Computing (AOC)*, ser. WoWMoM. IEEE, Jun. 2015.

[6] G. Blair, Y.-D. Bromberg, G. Coulson, Y. Elkhatib, L. Réveillère, H. B. Ribeiro, E. Rivière, and F. Taïani, "Holons: Towards a systematic approach to composing systems of systems," in *Workshop on Adaptive and Reflective Middleware (ARM)*. ACM, Dec. 2015.

[7] M. Siebert, P. Smits, L. Sauermann, and A. Dengel, "Increasing search quality with the semantic desktop in proposal development," in *Practical Aspects of Knowledge Management*. Springer, Nov. 2006, pp. 279–290.

[8] G. Denker, S. Nguyen, and A. Ton, "OWL-S semantics of security web services: a case study," in *European Semantic Web Symposium (ESWS)*. Springer, May 2004, pp. 240–253.

[9] A. Herzog, D. Jacobi, and A. Buchmann, "A3ME - an agent-based middleware approach for mixed mode environments," in *Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM)*, Sep. 2008, pp. 191–196.

[10] D. Preuveneers, J. V. den Bergh, D. Wagelaar, A. Georges, P. Rigole, T. Clerckx, Y. Berbers, K. Coninx, V. Jonckers, and K. D. Bosschere, "Towards an extensible context ontology for ambient intelligence," in *Ambient Intelligence*. Springer, 2004, pp. 148–159.

[11] M. Compton *et al.*, "The SSN ontology of the W3C semantic sensor network incubator group," *Web semantics: science, services and agents on the World Wide Web*, vol. 17, pp. 25–32, 2012.

[12] M. A. Musen *et al.*, "The Protégé Project: A look back and a look forward," *AI Matters*, vol. 1, no. 4, pp. 4–âĂŞ12, Jun. 2015.

[13] M. Milano and A. C. Myers, "MixT: A language for mixing consistency in geodistributed transactions," *SIGPLAN Not.*, vol. 53, no. 4, pp. 226–241, Jun. 2018.

[14] A. Jumagaliyev and Y. Elkhatib, "A modelling language to support evolution of multi-tenant cloud data architectures," in *Conference on Model Driven Engineering Languages and Systems (MODELS)*. ACM/IEEE, Sep. 2019.

[15] J. M. Sousa Fonseca, M. J. Varanda Pereira, and P. R. Henriques, "Converting ontologies into DSLs," in *Symposium on Languages, Applications and Technologies (SLATE)*, 2014.

[16] T. Walter, F. Silva Parreiras, and S. Staab, "Ontodsl: An ontology-based framework for domain-specific languages," in *Conference on Model Driven Engineering Languages and Systems (MODELS)*. ACM/IEEE, 2009.

[17] R. Agarwal, D. G. Fernandez, T. Elsaleh, A. Gyrard, J. Lanza, L. Sanchez, N. Georgantas, and V. Issarny, "Unified IoT ontology to enable interoperability and federation of testbeds," in *World Forum on Internet of Things (WF-IoT)*, Dec 2016, pp. 70–75.

[18] M. I. Ali, P. Patel, S. K. Datta, and A. Gyrard, "Multi-layer cross domain reasoning over distributed autonomous IoT applications," *Open Journal of Internet Of Things (OJIOT)*, vol. 3, no. 1, pp. 75–90, 2017.

[19] F. Kaddachi, H. Aloulou, B. Abdulrazak, P. Fraisse, and M. Mokhtari, "Technological approach for early and unobtrusive detection of possible health changes toward more effective treatment," in *Smart Homes and Health Telematics, Designing a Better Future: Urban Assisted Living*, M. Mokhtari, B. Abdulrazak, and H. Aloulou, Eds. Springer, 2018, pp. 47–59.

[20] H. Aloulou, M. Mokhtari, and B. Abdulrazak, "Deployment of an IoT solution for early behavior change detection," in *How AI Impacts Urban Living and Public Health*, J. Pagán, M. Mokhtari, H. Aloulou, B. Abdulrazak, and M. F. Cabrera, Eds. Springer, 2019, pp. 27–35.

[21] V. Nundloll, B. Porter, G. S. Blair, B. Emmett, J. Cosby, D. L. Jones, D. Chadwick, B. Winterbourn, P. Beattie, G. Dean, R. Shaw, W. Shelley, M. Brown, and I. Ullah, "The design and deployment of an end-to-end IoT infrastructure for the natural environment," *Future Internet*, vol. 11, no. 6, p. 129, 2019.

[22] L. M. Vaquero, F. Cuadrado, Y. Elkhatib, J. Bernal-Bernabe, S. N. Srirama, and M. F. Zhani, "Research challenges in nextgen service orchestration," *Future Generation Computer Systems*, vol. 90, pp. 20–38, Jan. 2019.

[23] A. E. Khaled and S. Helal, "Interoperable communication framework for bridging RESTful and topic-based communication in IoT," *Future Generation Computer Systems*, vol. 92, pp. 628 – 643, 2019.

[24] R. C. Gomez, Y.-D. Bromberg, Y. Elkhatib, L. Réveillère, and E. Rivière, "Emergent overlays for adaptive manet broadcast," in *International Symposium on Reliable Distributed Systems (SRDS)*. IEEE, Oct. 2019.

[25] M. Salama, Y. Elkhatib, and G. S. Blair, "IoTNetSim: A modelling and simulation platform for end-to-end IoT services and networking," in *Conference on Utility and Cloud Computing (UCC)*. ACM, 2019, pp. 251–261.