1-1-2019

Everything-as-a-Thing for Abstracting the Internet-of-Things

Zakaria Maamar
Zayed University

Noura Faci
Université Claude Bernard Lyon 1

Mohamed Sellami
ISEP Paris

Emir Ugljanin
State University of Novi Pazar

Ejub Kajan
State University of Novi Pazar

Everything-as-a-Thing for Abstracting the Internet-of-Things

Zakaria Maamar[1], Noura Faci[2], Mohamed Sellami[3], Emir Ugljanin[4] and Ejub Kajan[4]
[1]*Zayed University, Dubai, U.A.E.*
[2]*Université Lyon 1, Lyon, France*
[3]*ISEP Paris, Paris, France*
[4]*State University of Novi Pazar, Novi Pazar, Serbia*

Keywords: Capability, Internet of Things, Role, Service, Storytelling.

Abstract: This paper discusses Everything-as-a-Thing (*aaT) as a novel way for abstracting the Internet-of-Things (IoT) applications. Compared to other forms of abstraction like Everything-as-a-Service (*aaS) and Everything-as-a-Resource (*aaR), *aaT puts emphasis on living things, on top of non-living things, that populate these applications. On the one hand, living things take over roles that are defined in terms of rights and duties. On the other hand, non-living things offer capabilities that are defined in terms of functional and non-functional properties. Interactions that occur between living and non-living things are specified as stories that define who does what, when, and where. For illustration purposes, *aaT is put into action using a healthcare case study.

1 INTRODUCTION

A plethora of buzzwords (e.g., cloud, fog, ubiquitous, pervasive, and big data) describe the continuous development of the ICT field that is seeing, among other things, a convergence of system development approaches towards the paradigm of service computing. This paradigm is about exposing Everything-as-a-Service (*aaS). Thing could be software, platform, infrastructure, data communication, to cite just some, with the first 3 constituting the essence of cloud computing. *aaS offers ICT practitioners multiple advantages over other forms of computing (like component-based), such as, abstracting the complexity of the digital and physical worlds and their potential connection, complying with the separation-of-concerns principle (Parnas, 1972), and shifting the burden of running certain internal operations to external bodies in-return of a fee.

In conjunction with the expansion of service computing, the ICT field is, also, witnessing a phenomenal increase in the volume of data that is generated and, thus, needs to be "harnessed" in terms of processing and storage. Indeed, according to Vice President of Intel's Architecture Group, Kirk Skaugen, there was more data transmitted over the Internet in 2010 than the entire history of the Internet through 2009[1]. One source of these data are millions of *sensors* and *actuators* that perfectly exemplify the era of Internet-of-Things (IoT). IoT is about anything and everything (e.g., smartphone, kitchen appliance, and TV) that connects with peers and exchanges data with them. According to Gartner[2], 6.4 billion connected things (not to confuse with everything as a service) were in use in 2016, up 3% from 2015, and will reach 20.8 billion by 2020. In addition, it is predicted that the total economic impact of IoT will reach between $3.9 trillion and $11.1 trillion per year by the year 2025 (DZone, 2017).

In this paper, we discuss IoT from a service perspective by raising 2 specific questions: what is a thing when it acts as a consumer of services and what is a thing when it acts as a provider of services? Answering these 2 questions would raise another question, which is: is there room for Everything-as-a-Thing (*aaT) in the current ICT landscape? Contrarily to *aaS where everything is about non-living digital things, only, *aaT would include all forms of things, living and non-living. *aaT would, first, back Snyder and Byrd's vision about the Internet-of-Everything[3] that is the next evolution stage of IoT (Snyder and Byrd, 2017), and, second, respond to Moldovan et al.'s statements that "*Boundaries between computing systems, people, and things are gradually disappearing*" and that "*New approaches are required to manage today's and tomorrow's increa-*

[1]mashable.com/2011/10/20/kirk-skaugen-web-2.

[2]www.gartner.com/newsroom/id/3165317.

[3]Internet-of-Everything *versus* Internet-of-Things is discussed in (Simmons, 2015).

Maamar, Z., Faci, N., Sellami, M., Ugljanin, E. and Kajan, E.
Everything-as-a-Thing for Abstracting the Internet-of-Things.
DOI: 10.5220/0006869907330740
In *Proceedings of the 13th International Conference on Software Technologies (ICSOFT 2018)*, pages 733-740
ISBN: 978-989-758-320-9

singly connected and heterogenous ecosystems of people, computing processes, and things" (Moldovan et al., 2018).

Considering things as providers and/or consumers of services would require "revisiting" how today's things operate. Several reports indicate that things are still passive and mainly confined into a data-provider role (Green, 2015; Mzahm et al., 2013; Wu et al., 2014). In addition, and according to Gartner's hype cycle of emerging technologies (Snyder and Byrd, 2017), IoT has entered the trough of disillusionment, "*a period of uninspiring results compared to high expectation*". To fix this disillusionment and promote things from a data-provider role to service-provider/consumer role, we deem necessary examining how to support things "decide" on the best course of action to take in response to a particular surrounding. In this paper, we define course of action using storytelling concepts namely *character* and *script* (Ware et al., 2014). On the one hand, things are characters who either play roles of living things or offer capabilities of non-living things. On the other hand, things comply with scripts' what-to-do, when, and where.

Our contributions include (*i*) definition of IoT-related thing from a service perspective, (*ii*) empowerment of IoT-related thing so, that, a thing takes over/fulfills roles/capabilities, (*iii*) specification of IoT-related thing's roles/capabilities using storytelling, (*iv*) proposition of *aaT to abstract IoT applications development, and (*v*) illustration of *aaT through a healthcare case study. The rest of this paper is organized as follows. Section 2 consists of some definitions and then some related works. Section 3 presents *aaT in terms of foundations and how things bind to roles and capabilities. Section 4 concludes the paper.

2 SOME DEFINITIONS AND RELATED WORK

Storytelling. It has been used in different domains such as computer games and educational virtual environments. Storytelling has one main element, *story*, that features the following components (Young et al., 2013): (*i*) *script* that outlines a sequence and/or branching of actions and events related to the *story*, (*ii*) *characters* that set personalities along with their mental attitudes and relationships, and (*iii*) *settings* (*aka scenes*) that include spatio-temporal locations along with objects that *characters* manipulate when they join the *settings*. Details about using storytelling in game development are given in (Crawford, 2004), for example.

Internet-of-Things. The abundant literature on IoT (e.g., (Abdmeziem et al., 2016; Barnaghi and Sheth, 2016; DZone, 2017; Zorzi et al., 2010)) does not help propose a unique definition of what IoT is or should be. On the one hand, Barnaghi and Sheth provide a good overview of IoT requirements and challenges (Barnaghi and Sheth, 2016). 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 feature billions of dynamic things that make existing search, discovery, and access techniques and solutions inappropriate for IoT data and services. On the other hand, Abdmeziem et al. discuss IoT characteristics and enabling technologies (Abdmeziem et al., 2016). Characteristics include distribution, interoperability, scalability, resource scarcity, and security. And, enabling technologies include sensing, communication, and actuating. These technologies are mapped onto a three-layer IoT architecture that are referred to as perception, network, and application, respectively. Qin et al. (Qin et al., 2014) define IoT from a data perspective as *"In the context of the Internet, addressable and interconnected things, instead of humans, act as the main data producers, as well as the main data consumers. Computers will be able to learn and gain information and knowledge to solve real world problems directly with the data fed from things. As an ultimate goal, computers enabled by the Internet of Things technologies will be able to sense and react to the real world for humans"*.

Thanks to a reliable and efficient Internet, the Web has become the platform of choice for thousands of online transactions (related to e-commerce, e-government, e-learning, etc.) involving thousands of things exposed as services (*aaS). However, *aaS seems overlooking a major stakeholder in the equation of achieving these transactions, namely people who are now labeled as prosumers standing for consumers of services and providers of services (Pedrinaci and Domingue, 2010). *aaS does not capture the people dimension when exposing everything, including humans, as a service. Our *aaT addresses this limitation by expanding the list of things to people, who, on top of authorizing the use of their personal resources (e.g., desktops and networks), will have a say on how online transactions should be shaped due to concerns like privacy, limited availability, and social attitude. Thus, we consider *aaT as a normal evolution of *aaS and all its derivatives like *aaR (Resource) (Ba-

ker et al., 2018).

In (Christophe et al., 2011), the authors discuss the vision of a Web of things in which things are exposed as services and interactions with these services are defined as patterns. Though Christophe et al's work is a bit outdated, published in 2011, their vision has, nowadays, become a reality with the multitude of everyday objects connected to the Internet. To include humans in the list of everyday's "objects", *aaT offers the necessary means namely roles and capabilities to define living and non-living things, respectively.

In (Perera et al., 2014), sensing-as-a-service model is presented using 5 actors: sensors, owners of sensors, publishers of sensors, providers of sensed data, and consumers of sensed data. Benefits of using this model include embracement of cloud computing principles, participatory sensing, sharing and reusing sensor data, and fostering innovation. The model is evaluated through the win-win situation based on several objects that share common sensing data: a person with a new refrigerator with built-in sensors (temperature, door, etc.), a sensor publisher, an ice cream manufacturing company, and a cheese manufacturer. All of these objects expose themselves as sensors whose data are mutually shared in favor of all with respect to their roles, preferences, and capabilities.

In (Broring et al., 2017), the authors introduce an architectural model for IoT ecosystems and highlight 5 common interoperability patterns that would enable cross-platform interoperability among highly heterogeneous entities like providers and consumers from different application domains, among providers hosted on different IoT platforms, etc., and thus, establishing successful IoT ecosystems. A particular cross-platform pattern enables applications to access resources (information or functionality) from multiple platforms through the same interface specification. Despite the focus on interoperability across IoT applications, services and platforms, the idea of using patterns to allow multi-purpose of things may be extended to anything on the Web.

In (Chen et al., 2014), the authors propose Wisdom-as-a-Service (WaaS) model using 4 types of services: data, information, knowledge, and wisdom. WaaS provides intelligent IT applications based on a variety of intelligent technologies (e.g., personalization and context-awareness) for making judgments and taking actions so, that, the right services are provided to humans. After humans consuming service the process is return to things to start new loop. The cycle from raw data to the right services is called Wisdom Web of Things (W2T) processing cycle that according to Chen et al. *"... realizes the harmonious symbiosis of humans, computers, and things in the hyper-world"*. Our *aaT describes this symbiosis in an unified manner.

In (Raggett, 2015), the author stresses out the role of avatars in virtualizing the Web-of-Things (WoT). On top of connected devices that illustrate things, Raggett considers living things, which is in line with our *aaT, as well as other objects. Another use of avatar-based WoT architecture is discussed in (Mrissa et al., 2015). Despite the novel concept of using avatars, the human dimension is missing from the collaboration space that arises between things.

3 *aaT APPROACH

3.1 Foundations

In compliance with the storytelling principles (Section 2), we define *aaT using 2 main concepts (Fig. 1): character that would abstract IoT applications' future stakeholders and script that would abstract IoT applications' future operations (or course of action) that the stakeholders will execute. Depending on a script's definition (e.g., narrative description), a character either takes over a Role (R) that we define using *duties* and *rights* (Section 3.2) or fulfills a Capability (C) that we, also, define using *functional* and *non-functional* (*aka* QoS) attributes (Section 3.3). On the one hand, role targets living things (humans). On the other hand, capability targets non-living things and permits to cater to the needs and requirements of *aaS. By differentiating living/role from non-living/capability, *aaT complies with the separation-of-concerns principle since each has different objectives to achieve, different needs to satisfy, and different requirements to meet.

In Fig. 1, we note that (*i*) scripts regulate the operations (what-to-do) of characters ((n,m) cardinality), (*ii*) a role may request other capabilities in accordance with its rights ($(0,n)$ cardinality), (*ii*) a role may supervise other roles in accordance with its duties, as well ($(0,n)$ cardinality), and (*iii*) a capability may be composed of other capabilities in accordance with its functional attributes ($(0,n)$ cardinality). More details about the interactions between characters are provided in Section 3.4.

3.2 Binding Characters to Roles

Characters, who refer to living things, take over Roles (R) defined in terms of rights (r) to request and duties (d) to achieve. Rights and duties vary depending

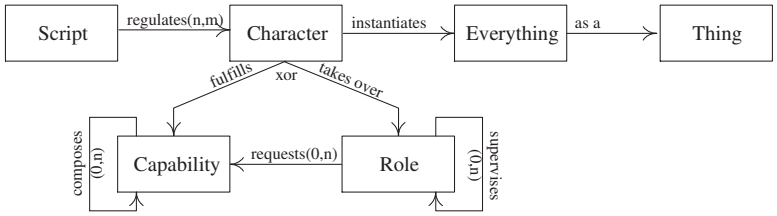

Figure 1: *aaT's core concepts for abstracting IoT applications' stakeholders/operations.

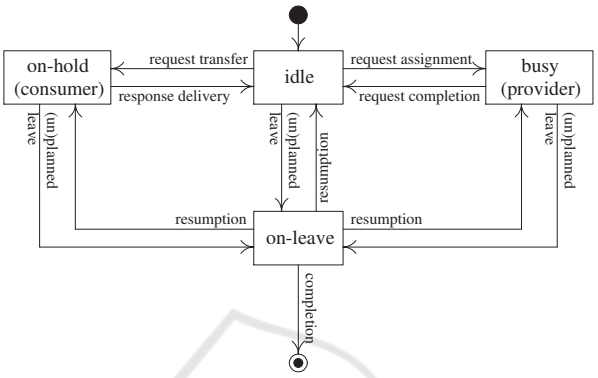

Figure 2: Lifecycle of a character taking over a role.

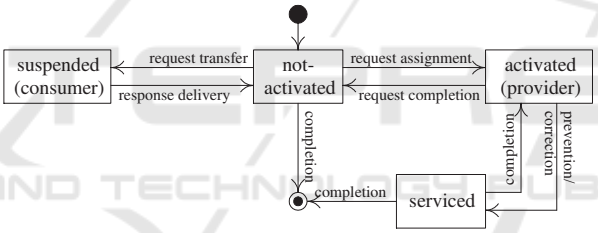

Figure 3: Lifecycle of a character fulfilling a capability.

Figure 4: Supervision cycle representing R2R interaction.

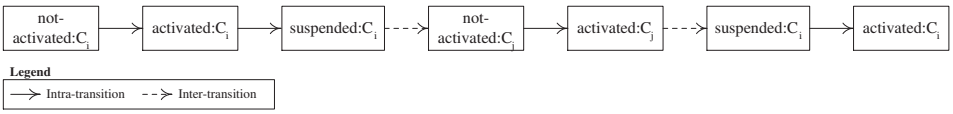

Figure 5: Composition cycle representing C2C interaction.

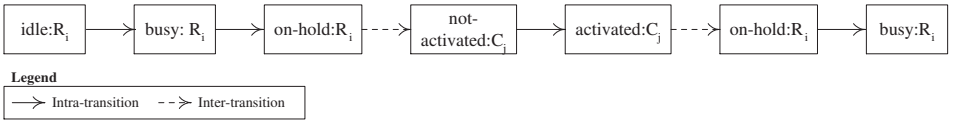

Figure 6: Request cycle representing R2C interaction.

on whether a character consumes services or provides services.

1. Character's rights/duties as a service consumer (c) are, but not limited to, listed below:
   - R.$c(r_1)$: search for other services provided by other characters.
   - R.$c(r_2)$: receive services from other characters according to what is agreed upon.
   - R.$c(r_3)$: raise concerns about other characters' behaviors.
   - R.$c(d_1)$: use valid credentials during identification.
   - R.$c(d_2)$: provide feedback, whenever necessary, on the consumed services of other characters.
   - R.$c(d_3)$: compensate the consumed services of other characters as agreed upon.
2. Character's rights/duties as a service provider (p) are, but not limited to, listed below:
   - R.$p(r_1)$: reject service requests (e.g., off-duty) of other characters.
   - R.$p(r_2)$: ask for the necessary capabilities when providing services to other characters.
   - R.$p(r_3)$: self-protect from malicious requestors of services.
   - R.$p(r_4)$: request fair treatment when competition arises.
   - R.$p(r_5)$: be compensated in return of the provided services to other characters.
   - R.$p(d_1)$: offer fair treatment to all requestors of services.
   - R.$p(d_2)$: allow any external request to audit (check) the provided services to other characters.

Building upon the aforementioned rights and duties, we illustrate in Fig. 2 a lifecycle for a character that takes over a role.

- As a service consumer, states include *idle*, *on-hold*, and *on-leave*. And, transitions include *request transfer*, *response delivery*, *completion*, *resumption* ($\times$ 2), and *(un)planned leave* ($\times$ 2).
- As a service provider, states include *idle*, *busy*, and *on-leave*. And, transitions include *request assignment*, *request completion*, *completion*, *resumption* ($\times$ 2), and *(un)planned leave* ($\times$ 2).

Since rights and duties are defined as abstract concepts, they need to be instantiated according to a case study, which is discussed in Section 3.5.

3.3 Binding Characters to Capabilities

Characters, that refer to non-living things, fulfill Capabilities (C) defined in terms of functional (f) attributes associated with what-is-offered and non-functional (nf) attributes associated with what-is-guaranteed. Functional and non-functional attributes vary depending on whether the character consumes services or provides services.

1. Character's functional/non-functional attributes as a service consumer (c) are, but not limited to, listed below:
   - C.$c(f_1)$: compose services provided by other characters according to what is agreed upon.
   - C.$c(nf_1)$: use valid credentials during authentication.
2. Character's functional/non-functional attributes as a service provider (p) are, but not limited to, listed below:
   - C.$p(f_1)$: announce services.
   - C.$p(f_2)$: monitor provided services.
   - C.$p(nf_1)$: ensure the quality-of-service of the provided services to other characters.
   - C.$p(nf_2)$: allow any external request to audit (check) the provided services to other characters.

Building upon the aforementioned functional and non-functional attributes, we illustrate in Fig. 3 a lifecycle for a character that fulfills a capability.

- As a service consumer, states include *not-activated* and *suspended*. And, transitions include *request transfer*, *response delivery*, and *completion*.
- As a service provider, states include *not-activated*, *activated*, and *serviced*. And, transitions include *request assignment*, *request completion*, *completion* ($\times$ 3), and *prevention/correction*.

Since functional and non-functional attributes are defined as abstract concepts, they need to be instantiated according to a case study, which is discussed in Section 3.5.

3.4 Interactions between Characters

Fig. 1 illustrates 3 types of interactions that involve characters together.

- Role-2-Role (R2R) interaction happens through *supervise*(0,n) relation. Acting as a provider of services, a character's role may[4] consist of looking after other characters (also acting as providers

[4]Because of 0 min in the cardinality.

of services) whose (certain) duties are required for achieving this role's duties.

For modeling needs of R2R interaction, we map *supervise* relation onto a supervision cycle that involves 2 roles, R_i:supervisor and R_j:supervisee, and is represented as a set of connected states that originate from these roles' respective lifecycles whether a role is a consumer (supervisor) or provider (supervisee). On top of these states, the supervision cycle features 2 types of transitions: intra-transitions connecting states that belong to the same lifecycle (already shown in Fig. 2) and inter-transitions connecting states that belong to separate lifecycles. Fig. 4 is the supervision cycle representing R2R interaction. Assumption made in this cycle is that the supervisee is waiting for requests from the supervisor. This cycle also highlights one intra-transition from *busy* to *on-hold* to allow the same supervisor to switch roles from provider to consumer and another intra-transition from *on-hold* to *busy* to allow the supervisor to switch roles from consumer to provider. These 2 intra-transitions need to be added to a character's lifecycle shown in Fig. 2.

- Capability-Capability (C2C) interaction happens through *compose*(0,*n*) relation. Acting as a provider of services, a character's capability may consist of initiating other characters (also acting as providers of services) whose (certain) capabilities are required for achieving this capability's functional attributes.

  For modeling needs of C2C interaction, we map *compose* relation onto a composition cycle that involves 2 capabilities, C_i:composer and C_j:component, and is represented as a set of connected states that originate from these capabilities' respective lifecycles whether a capability is a consumer (composer) or provider (component). On top of these states, the composition cycle features 2 types of transitions: intra-transitions connecting states that belong to the same lifecycle (already shown in Fig. 3) and inter-transitions connecting states that belong to separate lifecycles. Fig. 5 is the composition cycle representing C2C interaction. Assumption made in this cycle is that the component is waiting for requests from the composer. This cycle also highlights one intra-transition from *activated* to *suspended* to allow the composer to switch roles from provider to consumer and another intra-transition from *suspended* to *activated* to allow the same composer to switch roles from consumer to provider. These 2 intra-transitions need to be added to a character's lifecycle shown in Fig. 3.

- Role-Capability (R2C) interaction happens through *request*(0,*n*) relation. Acting as a provider of services, a character's role may consist of using other characters (also acting as providers of services) whose (certain) capabilities' functional attributes are required for achieving this role's duties.

  For modeling needs of R2C interaction, we map *request* relation onto a request cycle that involves 1 role, R_i:requestor, and 1 capability, C_j:requestee, and is represented as a set of connected states that originate from both this role's lifecycle acting as a consumer (requestor) and this capability's lifecycle acting as a provider (requestee). On top of these states, the request cycle features 2 types of transitions: intra-transitions connecting states that belong to the same lifecycle (already shown in Fig. 2 and Fig. 3) and inter-transitions connecting states that belong to separate lifecycles. Fig. 6 is the request cycle representing R2C interaction. Assumption made in this cycle is that the requestee is waiting for requests from the requestor. This cycle also highlights one intra-transition from *busy* to *on-hold* to allow the requestor to switch roles from provider to consumer and another intra-transition from *on-hold* to *busy* to allow the requestor to switch roles from consumer to provider. These 2 intra-transitions need to be added to a character's lifecycle shown in Fig. 2.

3.5 *aaT APPLICATION TO A CASE STUDY

To illustrate how we put *aaT into action (so that *aaT's concepts become concrete), a simple scenario is used. The scenario concerns a hospital that is on high-alert being close to a car accident. The hospital has different state-of-the-art equipment and facilities that showcase how IoT can smoothen operations and improve efficiency. For instance, wards have ambient sensors for temperature automatic-control, life-support machines have RFID tags for better tracking, and smart wrists allow real-time transmission of patients' vitals to appropriate recipients.

Let us consider an injured driver who requires a surgery due to brain bleeding. In compliance with Section 2's storytelling principles, the relevant *script* for surgery is activated as per the hospital's prescribed guidelines. A simple definition of this script[5] is given in Listing 1 where G stands for main goal, TG

[5]Script formalization does not fall into the scope of this work.

stands for terminal goal, CH stands for character, R stands for role, C stands for capability, T stands for time, and L stands for location. At run time, the script is activated, which means instantiating the necessary arguments. For instance, CH:R:doctor becomes John, CH:R:nurse becomes Melissa, CH:C:thermometer becomes thermometer$_{A12}$, T becomes 2:50pm, and L becomes operatingTheater$_{opt3}$.

Listing 1: Example of *script* definition.

```
Script = G:stop-brain-bleeding
          G:prepare-patient
           (TG:diagnose-patient;
             CH:R:nurse.takeTemp.CH:R:patient OR CH:C:
        thermometer.takeTemp.CH:R:patient,
             CH:R:nurse.takePress.CH:R:patient OR CH:C
        :smartWrist.takePress.CH:R:patient,
             CH:R:doctor.diagnose.CH:R:patient; T:
        currentTime; L:operatingTheater)
           (TG:give-medication; ..... )
          G:perform-surgery
           (TG:prepare-operating-theater; .....)
           (TG:.....)
           .....
```

In Fig. 1, *aaT defines role with rights and duties, capability with functional and non-functional attributes, and, finally, relations between roles, between capabilities, and between roles and capabilities. We illustrate all these concepts with the brain bleeding surgery.

Doctor Role: to identify the duties and rights of doctor as role, we resort to job descriptions that clearly define these duties and rights from different perspectives like patient, line-manager, peer, community, etc. In the following, we consider the patient perspective.

Examples of Doctor's rights/duties as a service consumer (c) are:

- Doctor.$c(r_1)$: consult patient files prior and after surgeries.
- Doctor.$c(r_2)$: have access to surgery equipment.
- Doctor.$c(d_1)$: return patient files after surgeries..
- Doctor.$c(d_2)$: meet deadlines when submitting surgery requests.

Examples of Doctor's rights/duties as a service provider (p) are:

- Doctor.$p(r_1)$: inform insurance providers about patient conditions before and after surgeries.
- Doctor.$p(r_2)$: postpone patient surgeries.
- Doctor.$p(d_1)$: consult patients before and after surgeries.
- Doctor.$p(d_2)$: prescribe medicines before and after surgeries.

Thermometer Capability: to identify the functional and non-functional attributes of thermometer as capability, we resort to the descriptions that a maker of this thermometer provides.

Examples of Thermometer's functional/non-functional attributes as a service consumer (c) are:

- Thermometer.$c(f_1)$: not-applicable.
- Thermometer.$c(nf_1)$: use 2mm diameter disposable lens-filters.

Examples of Thermometer's functional/non-functional attributes as a service provider (p) are:

- Thermometer.$p(f_1)$: measure body temperature.
- Thermometer.$p(nf_1)$: measure body temperature with 99% accuracy.

Doctor-Nurse Relation: to illustrate the *supervise* relation between doctor and nurse roles, we stress out the duties of doctor that call for the duties of nurse. Doctor's duties will be listed from a service-consumer perspective whereas nurse's duties will be listed from a service-provider perspective. Examples of duties are:

- Doctor.$c(d_1)$: request patient vitals from nurse.
- Nurse.$p(d_1)$: respond to doctor with patient vitals.

smartWrist-Thermometer Relation: to illustrate the *compose* relation between smartWrist and thermometer capabilities, we stress out the capabilities of smartWrist that call for the capabilities of thermometer. smartWrist's and thermometer's capabilities will be listed from a service-provider perspective. Examples of capabilities are:

- Thermometer.$p(f_1)$: measure body temperature.
- smartWrist.$p(f_1)$: inform nurse based on the body temperature returned by thermometer.

Nurse-Thermometer Relation: to illustrate the *request* relation between nurse as role and thermometer as capability, we stress out the duties of nurse that call for the capabilities of thermometer. Nurse's duties will be listed from a service-consumer perspective whereas thermometer's capabilities will be listed from a service-provider perspective. Examples of duties and capabilities are:

- Nurse.$c(d_1)$: request body temperature from thermometer.

- Thermometer.$p(f_1)$: measure body temperature.

4 CONCLUSION

This paper presented Everything-as-a-Thing (*aaT) as a new paradigm for abstracting the Internet-of-Things (IoT). Compared to Everything-as-a-Service (*aaS) and Everything-as-a-Resource (*aaR), *aaT differentiates living from non-living things. The former take over roles that are defined in terms of rights and duties, and, the latter offer capabilities that are defined in terms of functional and non-functional properties. *aaT, also, relies on storytelling's principles, namely script to define what living and non-living things are expected to perform and character to bind living and non-living things to specific scripts. Our ongoing work consists of applying *aaT to a real scenario like the one discussed in this paper.

REFERENCES

Abdmeziem, M. R., Tandjaoui, D., and Romdhani, I. (2016). *Architecting the Internet of Things: State of the Art*, pages 55–75. Springer International Publishing.

Baker, T., Ugljanin, E., Faci, N., Sellami, M., Maamar, Z., and Kajan, E. (2018). Everything as a Resource: Foundations and Illustration through Internet-of-Things. *Computers in Industry*, 94.

Barnaghi, P. M. and Sheth, A. P. (2016). On Searching the Internet of Things: Requirements and Challenges. *IEEE Intelligent Systems*, 31(6).

Broring, A., Schmid, S., Schindhelm, C., Khelil, A., Kabisch, S., Kramer, D., Phuoc, D. L., Mitic, J., Anicic, D., and Teniente, E. (2017). Enabling iot ecosystems through platform interoperability. *IEEE Software*, 34(1):54–61.

Chen, J., Ma, J., Zhong, N., Yao, Y., Liu, J., Huang, R., Li, W., Huang, Z., Gao, Y., and Cao, J. (2014). WaaS: Wisdom as a Service. *IEEE Intelligent Systems*, 29(6).

Christophe, B., Boussard, M., Lu, M., Pastor, A., and Toubiana, V. (2011). The web of things vision: Things as a service and interaction patterns. *Bell Labs Technical Journal*, 16(1):55–61.

Crawford, C. (2004). *Chris Crawford on Interactive Storytelling (New Riders Games)*. New Riders Games, CA, USA.

DZone (https://dzone.com/guides/iot-applications-protocols-and-best-practices, 2017 (visited in May 2017)). The Internet of Things, Application, Protocols, and Best Practices. Technical report, DZone.

Green, H. (December 2015). The Internet of Things in the Cognitive Era: Realizing the Future and Full Potential of Connected Devices. www-01.ibm.com/common/ssi/cgi-bin/ssialias?htmlfid=WWW12366USEN.

Moldovan, D., Copil, S., and Dustdar, S. (2018). Elastic Systems: Towards Cyber-Physical Ecosystems of People, Processes, and Things. *Computer Standards & Interfaces*, 57.

Mrissa, M., Médini, L., Jamont, J., Le Sommer, N., and Laplace, J. (2015). An Avatar Architecture for the Web of Things. *IEEE Internet Computing*, 19(2).

Mzahm, A., Ahmad, M., and Tang, A. (2013). Agents of Things (AoT): An intelligent operational concept of the Internet of Things (IoT). In *Proceedings of the 13th International Conference on Intellient Systems Design and Applications (ISDA'2013)*, Bangi, Malaysia.

Parnas, D. (1972). On the Criteria to be Used in Decomposing Systems into Modules. *Communications of the ACM*, 15(12).

Pedrinaci, C. and Domingue, J. (2010). Toward the Next Wave of Services: Linked Services for the Web Data. *Journal of Universal Computer Science*, 16(13).

Perera, C., Zaslavsky, A., Christen, P., and Georgakopoulos, D. (2014). Sensing as a Service Model for Smart Cities Supported by Internet of Things. *Transactions on Emerging Telecommunications Technologies*, 25(1).

Qin, Y., Sheng, Q., Falkner, N., Dustdar, S., Wang, H., and Vasilakos, A. (2014). When Things Matter: A Data-Centric View of the Internet of Things. *CoRR*, abs/1407.2704.

Raggett, D. (2015). The Web of Things: Challenges and Opportunities. *Computer*, 48(5).

Simmons, L. ((last checked out March 2018) October 2015). What is the Difference between the Internet-of-Everything and the Internet-of-Things? blog.cloudrail.com/internet-of-everything-vs-internet-of-things.

Snyder, T. and Byrd, G. (June 2017). The Internet of Everything. *Computer*, 50(6).

Ware, S., Young, R., Harrison, B., and Roberts, D. (2014). [a computational model of narrative conflict at the fabula level. *IEEE Transactions on Computational Intelligence and Artificial Intelligence in Games*, 6(3).

Wu, Q., Ding, G., Xu, Y., Feg, S., Du, Z., Wang, J., and Long, K. (April 2014). Cognitive Internet of Things: A New Paradigm Beyond Connection. *IEEE Internet of Things Journal*, 1(2).

Young, R. M., Ware, S. G., Cassell, B. A., and Robertson, J. (2013). Plans and Planning in Narrative Generation: A Review of Plan-based Approaches to the Generation of Story, Discourse and Interactivity in Narratives. *Sprache und Datenverarbeitung, Special Issue on Formal and Computational Models of Narrative*, 37(1-2).

Zorzi, M., Gluhak, A., Lange, S., and Bassi, A. (2010). From Today's INTRAnet of Things to a Future INTERnet of Things: a Wireless- and Mobility-related View. *IEEE Wireless Commun.*, 17(6).