Service Oriented Computing and Applications (2024) 18:15–31
https://doi.org/10.1007/s11761-023-00379-3

ORIGINAL RESEARCH PAPER

Thing Artifact-based Design of IoT Ecosystems

Zakaria Maamar[1] · Noura Faci[2] · Mohammed Al-Khafajiy[3] · Murtada Dohan[4]

Received: 2 September 2023 / Revised: 2 November 2023 / Accepted: 11 November 2023 / Published online: 11 December 2023

Abstract
This paper sheds light on the complexity of designing Internet of Things (IoT) ecosystems where a high number of things reside and thus must collaborate despite their reduced size, restricted connectivity, and constrained storage limitations. To address this complexity, a novel concept referred to as thing artifact is devised abstracting the roles that things play in an IoT ecosystem. The abstraction focuses on 3 crosscutting aspects, namely functionality in terms of what to perform, life cycle in terms of how to behave, and interaction flow in terms of with whom to exchange. Building upon the concept of data artifact commonly used in data-driven business applications design, thing artifacts engage in relations with peers to coordinate their individual behaviors and hence avoid conflicts that could result from the quality of exchanged data. Putting functionality, life cycle, interaction flow, and relation together contributes to abstracting IoT ecosystems design. A system implementing a thing artifact-based IoT ecosystem along with some experiments is presented in the paper as well.

Keywords Data artifact · Data quality · IoT ecosystem · Thing artifact

1 Introduction

It is largely accepted that organizations rely on business processes (BPs) to achieve goals, sustain growth, improve services, and many other things that could keep them competitive. In the Information and Communication Technology (ICT) community, 2 major trends related to BPs design exist: activity-centric and data-centric. The activity-centric trend produces process models that define the execution chronology of BPs' activities along with who does what, where, when, and why (5W's) [24]. And, the data-centric trend produces Data Artifacts (DAs) that define BPs' data and events along with the changes that these data are subject to because of these events [17]. Independently of how these 2 trends examine BPs design since the activity-centric trend is about what should be done and the data-centric trend is about what can be done [11], a BP remains "*the coding of a lesson learnt in the past, transformed into a standard by a group of experts, and established as a mandatory flow for those who must effectively carry out the work*" [18].

In addition to BPs, organizations rely on other ICTs like lately Blockchain, data analytics, cloud/edge computing, and Internet of Things (IoT) that is the focus of this paper. IoT is about anything and everything (e.g., smartphones, kitchen appliances, and embedded systems) that connect with other things to enable control of users' cyber-physical surroundings, for example. According to Techjury,[1] 25 billion connected things were in use in 2021 and will reach 64 billion by 2025. All these things are feeding computers with data to address real-world problems and are also sensing and reacting to the real world of humans [21]. It is predicted that the total economic impact of IoT will reach between $3.9 trillion and $11.1 trillion per year by 2025 [8].

According to Nigam and Caswell [17], a DA is a concrete, identifiable, self-describing chunk of information that can be used by a business person to actually run a business, and has a set of states that constitute the DA's life cycle. Should we apply Nigam and Caswell's DA definition to IoT, then questions like could DAs help identify necessary things in an IoT ecosystem, could DAs capture things' operations and interactions, and could DAs support things engage in collaborative scenarios will require responses from the ICT community. In fact, could we expect a generation of

✉ Zakaria Maamar
zakaria.maamar@udst.edu.qa

1 University of Doha for Science and Technology, Doha, Qatar

2 Université Lyon 1, CNRS, Liris, Lyon, France

3 University of Lincoln, Lincoln, UK

4 University of Northampton, Northampton, UK

[1] www.techjury.net/blog/internet-of-things-statistics.

Thing Artifacts (TAs) that will be built upon DAs to design and develop IoT ecosystems? By analogy with DAs, we foresee a TA as a chunk of information capturing 3 crosscutting aspects, namely functionality, life cycle, and interaction flow. These 3 aspects capture what, how, and with whom, respectively. Indeed, functionality is what a TA does, so an IoT ecosystem functions like monitoring ambient temperatures. Life cycle is how a TA behaves when it provisions its functionality to the IoT ecosystem. Finally, interaction flow is with whom a TA exchanges messages while its life cycle is activated. Although some specific aspects like longevity and security could be considered, functionality, life cycle, and interaction aspects are comprehensive providing a complete TA definition.

To ensure free of conflicts TAs collaboration due to potential conflicting data, for example, we tap into the latest trend of blending social computing with IoT leading to what is commonly known as Social Internet of Things (SIoT, [4, 15, 20]). Relations like parental, co-working, and ownership capturing real-life situations between humans are applied to things. Could we apply the same relations to TAs based on their respective functionalities?

In this paper our contributions are as follows

1. Definition of TA for IoT ecosystems design. This definition checks existing works on DAs use for BPs design.
2. Identification of social relations between TAs residing in an IoT ecosystem. This identification checks existing works on having things participate in social relations.
3. Analysis of data quality impact on TAs collaboration. This analysis specializes data quality into uncertainty, redundancy, and inconsistency and defines the impact of each on future social relations between TAs.
4. And, development of a system demonstrating and evaluating TAs put into action as part of a complete IoT ecosystem.

The rest of this paper is organized as follows. Section 2 defines the key concepts of IoT and DA. Sect. 3 suggests a case study motivating both IoT and TAs adoption. Sect. 4 is related to TA-based design of IoT ecosystems. The impact of data quality on TAs collaboration is discussed in Sect. 5. Technical details about TAs implementation are given in Sect. 6. Finally, Sect. 7 concludes the paper and discusses future work.

2 Background

After a brief overview of IoT, the rest of the section is dedicated to DAs shedding light on some relevant features that could cater for the needs of IoT ecosystems design and thus motivate the development of TAs.

Internet of Things. A good amount of works on IoT exist in the literature, which in fact does not help come up with a unique definition of IoT (e.g., [1], [5], [23], and [26]). For instance, Abdmeziem et al. discuss IoT characteristics and enabling technologies [1]. Characteristics include distribution, interoperability, scalability, resource scarcity, and security. And, enabling technologies include sensing, communication, and actuating and are mapped onto a three-layer IoT architecture referred to as perception, network, and application. In [5], Barnaghi and Sheth identify IoT's requirements and challenges. 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 featuring billions of dynamic things that make existing search, discovery, and access techniques inappropriate for IoT data and services. Finally, Qin et al. 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"* [21].

Data artifacts. Many benefits could be achieved through the use of DAs for instance, reducing process model complexity through the identification of DAs whose behaviors are modeled as life cycles and tracking process model execution as a set of communicating DAs' life cycles. A relevant illustration of a DA-centric approach is given in [7] with IBM Global Financing (IGF). The initial adoption of DAs such as deal, supplier invoice, and asset made IGF people grasp the complexity of the BPs that they were in charge of and thus were able to put forward proposals to streamline the progress of these processes according to their needs and requirements.

A good number of initiatives on DAs are reported in the literature where their discovery remains a critical factor to their successful adoption. This factor is stressed out by Bhattacharya et al. who developed a data-centric design method for BPs [6]. Quoted from Bhattacharya et al.'s paper, "*... Identifying artifacts require an understanding of the whole business process, how data are changed and shared through the process, and what data hold critical business process information.*" In [11], Kumaran et al. suggest guidelines to help IT practitioners identify the necessary DAs for their applications. The authors define DAs as dominant information entities having both (*i*) a data model that describes data dependencies between the dominant entity and dominated entities and (*ii*) a behavior that is modeled as a state machine where state transitions are caused by activities acting on the dominant entity. In [16], Narendra et al. use context-

based artifacts and Web services to model BPs. The authors abstract processes using models that are expressive (i.e., easy to "grasp") to non-IT practitioners and could be based on DAs. In [12], Maamar et al. discover and model DAs based on a set of business requirements. They use a bottom-up analysis to determine fine-grained data, which are afterward aggregated into clusters where each cluster becomes a potential DA. Next, they derive the operations that act upon the discovered data clusters. At the end, the data and operation clusters are grouped into final DAs. Maamar et al.'s discovery and modeling framework uses many concepts like dependency types, big artifact, sub-artifact, final artifact, data space, and operation space. In [19], Popova et al. acknowledge the role of DAs in modeling BPs and propose techniques to discover DAs' life cycles. These techniques are implemented as software plug-ins for ProM (www.promtools.org), a generic open-source framework for supporting process mining techniques.

3 Case study

To illustrate the blend of IoT with BPs from a data-centric perspective and hence, identify TAs that would contribute to IoT ecosystems design, our case study targets elderly people in a healthcare facility. Many studies confirm that population aging is a dominant global demographic trend of the twenty-first century.[2] In addition, many studies confirm the benefits of IoT for elderly people for instance *IoT for seniors: Solutions and Use Cases* by Cogniteq[3] and *How can IoT help with elderly care?* by Telefònica Tech.[4]

Let us start with an ecosystem that would revolve around the living room in the healthcare facility. An immediate situation requiring design would expose things like remote control, smart TV, and light switches. For a successful design so that these things would allow watching movies in an enjoyable atmosphere by automatically dimming the lights and selecting the best sound effects, a first step would consist of identifying what each thing does, how and why a thing acts upon the data it manipulates, how and when a thing interacts with peers, and how a thing responds to peers' messages. While Table 1 provides a sample of operations that some things would perform, a TA-based IoT ecosystem design would privilege data over operations like shown in this table's data column. Besides the data to associate with a TA's functionality, a TA will have a life cycle for its behavior and a set of interaction flows for the messages that it sends and receives.

In the same living room, another situation could arise involving this time a smart watch that synchronizes with a medical dispenser the automatic release of medicine doses to the concerned person. The synchronization will be set according to the treatment's duration but, then, extended, should the treatment be renewed. To accommodate this synchronization and extension during an IoT ecosystem design, TAs are associated with interactions flows and life cycles. First, interaction flows track how TAs respond to their peers' demands for instance, the smart watch triggering the medical dispenser. And, life cycles include states that reflect what TAs are doing for instance, the medical dispenser requesting refill because of medical treatment's extension.

Whether things or TAs, the 2 situations above offer a glimpse of potential relations in which things/TAs could participate. On the one hand, some things/TAs work together like smart watch alerting the medical dispenser about the time of dispensing pills. On the other hand, other things/TAs like smart watch and smartphone compete when both act as alarms. We depend on such relations during IoT ecosystems design.

4 IoT ecosystems design

To illustrate IoT ecosystems design based on TAs, we define first, functionality of thing, TA, relations between TAs, and, finally, interactions between TAs.

4.1 Functionality of thing

Building upon our previous work [13], functionality is what a thing does and is specialized into primitive and composite. From an abstract perspective, a primitive functionality could be either *sensing* (collecting data), *actuating* (processing data), or *communicating* (distributing data). Although this does not fall into this paper's scope, a functionality could be described with properties like frequency and availability of sensing, quality and accuracy of actuating, and reliability and responsiveness of communicating. As per Fig. 2, a functionality is either disabled or enabled (0,1) according to the functional/non-functional requirements of the underdevelopment IoT application to integrate into the IoT ecosystem. Briefly, a thing senses the cyber-physical surrounding, so that it generates (raw) data. A thing actuates data including those that are sensed. Finally, a thing communicates with the cyber-physical surrounding the sensed and/or actuated data. With reference to the case study, we abstract the light switch with an *actuating* turning the lights on and off.

A composite functionality puts primitive functionalities together according to specific cases since it is unlikely that one thing alone would cater for the needs of today's IoT applications. These cases are *sensing.actuating.communicating*

[2] www.weforum.org/agenda/2019/10/ageing-economics-population-health.

[3] www.cogniteq.com/blog/iot-seniors-solutions-and-use-cases.

[4] www.telefonicatech.com/en/blog/how-can-iot-help-with-elderly-care.

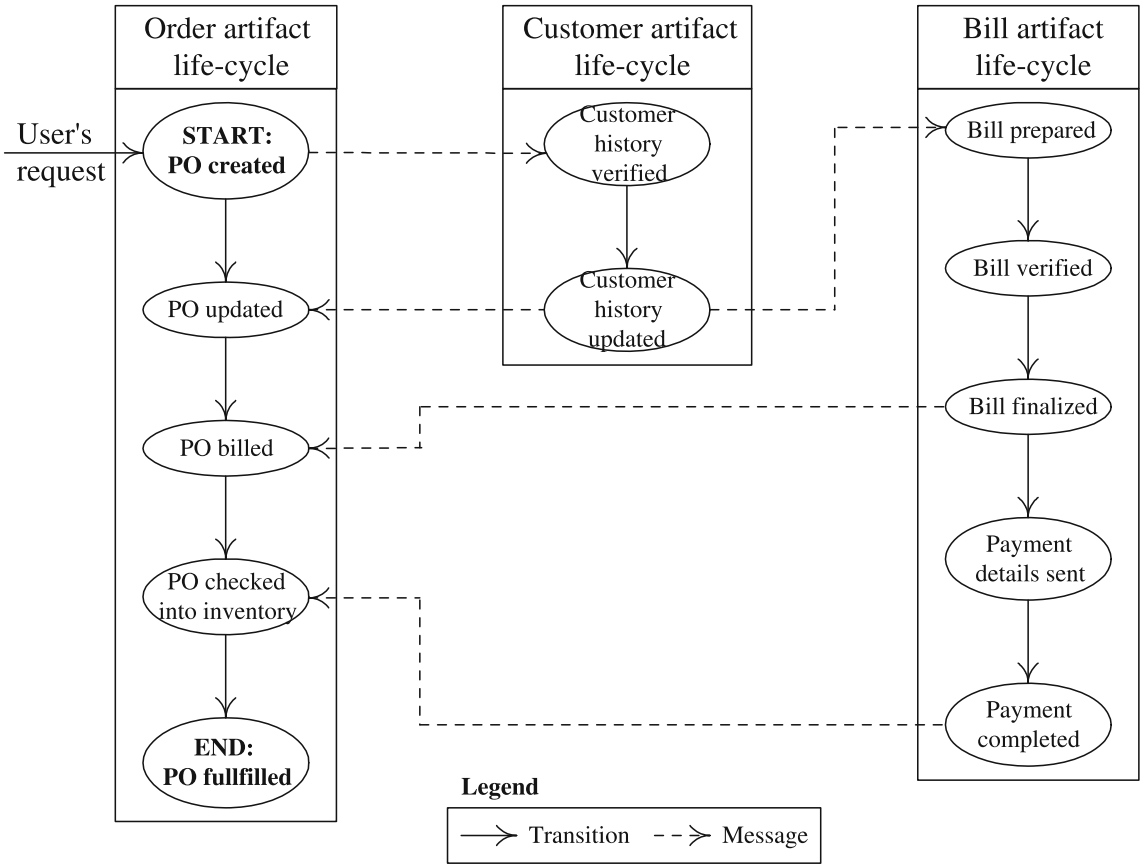


Fig. 1 Representation of a DA-based purchase order BP (adapted from [16])

Table 1 Thing/Operation *versus* TA/Data in an IoT ecosystem

Thing	Some operations to perform	TA	Some data to capture
Smart TV	display (enable programs for viewing), record (tape programs), share (send the cable TV company some details), etc.	Smart TV	size of the TV, name of the program, rating of the program, producer of the program, brightness level, etc.
Medical dispenser	change (modify the intake frequency per type of pill), configure (select the container per type of pill), display (notify availability of pills on the TV screen), etc.	Medical dispenser	name of medicine, dosage of medicine, frequency of medicine, side effects of medicine, etc.

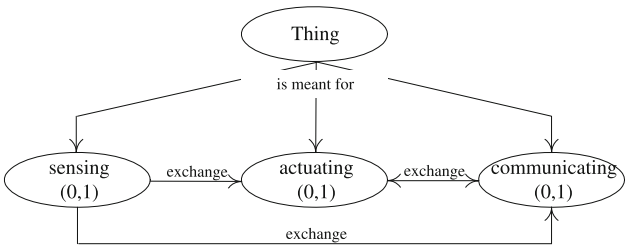


Fig. 2 Classification of a thing's primitive functionalities

(sensed data are passed on to actuating; and the data that result from actuating are passed on to communicating for distribution), *sensing.actuating* (sensed data are passed on to actuating; and the data that result from actuating are finals), *sensing.communicating* (sensed data are passed on to communicating for distribution), *actuating.communicating* (data that result from actuating are passed on to communicating for distribution), and *communicating.actuating* (data that result from communicating are passed on to actuating). It should be noted that sequences like *sensing.actuating.sensing* are not valid from an operational perspective and thus are dropped

from our analysis. With reference to the case study, we abstract the remote control with *actuating.communicating* controlling the smart TV and requesting the switch to turn the lights off, if required.

4.2 Definition of thing artifact

In line with Qin et al. who state that processing data in IoT is different from traditional Internet environments [21], we argue that a data-centric perspective would make data a "first-class citizen" during IoT ecosystems design following the blend of IoT with BPs. First of all, BPs' activities act upon data in compliance with create–read–update–delete (CRUD) operations. Contrarily, things act upon data in compliance with their sensing, actuating, and communicating functionalities. By having both BPs' activities and things' functionalities act upon data, we define a TA as a chunk of information that refers to its functionality, captures its life cycle, and tracks its interaction flows. Functionality is what a TA does; life cycle is how a TA behaves; and interaction flow is with whom a TA exchanges messages.

Definition 4.1 A TA is defined by the tuple $< f, D, lc, IF >$ where:

- f is either a primitive functionality, f_p, or a composite functionality, f_c, that the TA offers as per Sect. 4.1.
- D is a set of input data $\{d_i^{in}\}$ and output data $\{d_j^{out}\}$ that the TA's functionality acts upon using CRUD operations.
- lc is the TA's life cycle represented as a state diagram as per Definition 4.2.
- IF is a set of interaction flows, $\{if_i\}$, that are put together on the fly based on the messages that the TA sends and receives as per Definition 4.3.

Definition 4.2 A TA's life cycle, lc, is defined by the couple $< S, T >$, i.e., $lc = s_i \xrightarrow{trans_i} s_{i+1} \xrightarrow{trans_{i+1}} s_{i+2} \ldots s_{j-1} \xrightarrow{trans_{j-1}} s_j$, where:

- S is a set of states $\{s_i\}$ as per Fig. 3. In S, an *activated* state encompasses other sub-states (dashed lines) that capture the TA's type of functionality whether primitive (sensing *versus* actuating *versus* communicating) or composite (sensing.communicating *versus* communicating.actuating.sensing *versus* ...).
- T is a set of transitions $\{trans_i\}$ as per Fig. 3.

By analogy with Fig. 1 where order, customer, and bill DAs are each associated with a life cycle, TAs will have the same capturing their behaviors at run-time. Represented as a state diagram, a behavior corresponds to 5 states, *prepared*, *activated*, *done*, *suspended*, and *failed*. And, transiting from one state to another is dependent on events like temperature drop and/or messages exchanged between TAs.

Definition 4.3 A TA's interaction flow, if_i, is a set of messages $\{m_{ij}\}$ that the TA uses to interact with peers. A message is defined by the tuple $< id, type, from/state, to/state, cnt >$ where:

- id is a message's identifier.
- $type$ is a message's type belonging to a list of predefined terms defined in Table 4.
- $from/state$ is the sending state in the TA's life cycle.
- $to/state$ is the receiving state in the life cycle of a peer to the TA.
- cnt is a content of data conveyed between the TA and a peer where $cnt \subseteq D$ as per Definition 4.1.

4.3 Relations between thing artifacts

In [9], Ghajargar et al. analyze and design IoT artifacts using *augment-me*, *comply-with-me*, *engage-me*, and *make-me-think* relations. These latter connect IoT artifacts to users, tasks that users initiate, and situations in which users reside. For illustration, Fig. 4 shows *augment-me* where there is a balanced collaboration between the user and artifact that is a calculator, there is always a specific task that the artifact will perform, and the user has full control of the relation.

Building upon Ghajargar et al.'s 4 relations and some project management scheduling techniques (Table 2 where f, sf, fs, ss, and ff stand for functionality, start-to-finish, finish-to-start, start-to-start, and finish-to-finish, respectively), and assuming that each TA would have an operation time interval, $[b, e]$ (where b/e stands for begin/end), we propose *work-with-me*, *work-for-me*, *back-me*, and *avoid-me* relations that would define the interaction flows between an IoT ecosystem's TAs. These relations are set according to TAs' functionalities and are analyzed from a temporal perspective using Allen's time algebra ([2], Table 3).

Using Table 2, we specify TAs' operations in an IoT ecosystem. After selecting a particular scheduling technique like *start-to-start*, we set the values of the TAs' operation time intervals according to the underdevelopment IoT application's requirements. Then, we define appropriate time relations between these intervals using Allen's time algebra and the specificities of each relation as per the discussion below:

- *Work-with-me* (Fig. 5): it has an initiator TA_i and a set of collaborating partners $1[\mathsf{TA}_j]n$ (where $1[..]n$ stands for one-to-many) with whom the initiator collaborates ① to provision a *composite functionality* ② to the IoT ecosystem ③. *Work-with-me*'s characteristics include peer-to-peer collaboration between TA_i and $1[\mathsf{TA}_j]n$, complementary functionalities between all TAs ($\mathsf{TA}_i.f \cap$

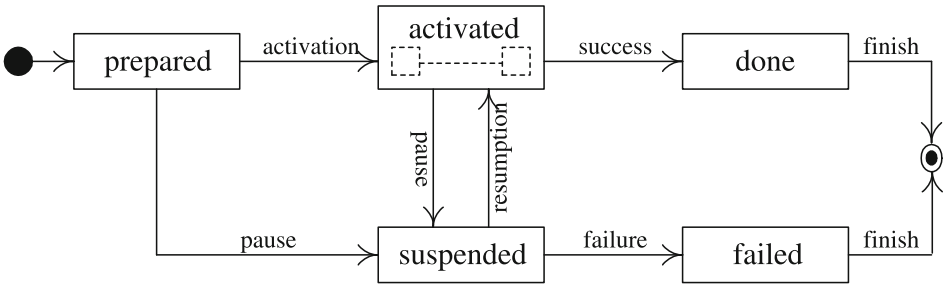


Fig. 3 TA's life cycle as a state diagram

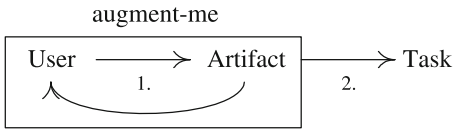


Fig. 4 Representation of *augment-me* relation ([9])

$1[\mathsf{TA}_j.f]n = \phi$), and TAs synchronization so they are all active while the relation lasts. Let $\mathsf{TA}_{1,2,3}$ be smart TV, remote control, and light switch, respectively. We exemplify *work-with-me* between TA_1 and $[\mathsf{TA}_2,\mathsf{TA}_3]$ with a viewing session where the smart TV would record (i.e., $actuating_1$) a movie's title, rating, and duration. During advertisements, the smart TV communicates the required level of brightness (i.e., *communicating*) to the remote control that, in turn, communicates (i.e., *communicating*) the new level to the light switch. As a result, the light switch adjusts the living room's brightness and the smart TV is muted (i.e., $actuating_2$). Because *work-with-me* requires the active participation of all the TAs, synchronizing them would refer to specific scheduling techniques that, themselves, would refer to specific time relations between their respective operation time intervals as discussed below using TA_i and TA_j:

1. *ss*(TA_i,TA_j): as both TA_i and TA_j would start together, there will be one possible time relation between their respective operation time intervals that is
   (a) *equals*($\mathsf{TA}_i[b_i,e_i]$,$\mathsf{TA}_j[b_j,e_j]$) where $b_i = b_j$ and $e_i = e_j$.
2. *ff*(TA_i,TA_j): as both TA_i and TA_j would finish together, there will be one possible time relation between their respective operation time intervals that is
   (a) *equals*($\mathsf{TA}_i[b_i,e_i]$,$\mathsf{TA}_j[b_j,e_j]$) where $b_i = b_j$ and $e_i = e_j$.

- *Work-for-me* (Fig. 6): it has an initiator TA_i and a set of partners $0[\mathsf{TA}_j]n$ (where $0[...]n$ stands for zero-to-many in the sense that partners $[\mathsf{TA}_j]$ might not "accept" working for TA_i, which is not the case in *work-with-me* where a composite functionality needs to be provisioned) that are asked by the initiator to provision its functionality ① to the IoT ecosystem ②. *Work-for-me*'s characteristics would include master–slave collaboration between TA_i and $0[\mathsf{TA}_j]n$, common functionalities between all TAs ($\mathsf{TA}_i.f \cap 0[\mathsf{TA}_j.f]n \neq \phi$), and synchronization of all TAs to ensure the existence of a time frame (at least once) where they are all active while the relation lasts. Let $\mathsf{TA}_{1,2}$ be an elderly person's smart phone and smart TV, respectively. We exemplify *work-for-me* between TA_1 and TA_2 with a physiotherapy session where the smart phone should continuously play (i.e., *actuating*) music but, due to incoming calls, the smart phone streams the music through the smart TV's speakers (i.e., *actuating*) during this session. Because *work-for-me* requires the active (but not necessarily continuous) participation of all the TAs, synchronizing them would refer to specific scheduling techniques that, themselves, would refer to specific time relations between their respective operation time intervals as discussed below using TA_i and TA_j:

  1. *ss*(TA_i,TA_j): as both TA_i and TA_j would start together, there will be one possible time relation between their respective operation time intervals that is
     (a) *equals*($\mathsf{TA}_i[b_i,e_i]$,$\mathsf{TA}_j[b_j,e_j]$) where $b_i = b_j$ and $e_i = e_j$.
     (b) *starts*($\mathsf{TA}_i[b_i,..]$,$\mathsf{TA}_j[b_j,..]$) where $b_i = b_j$.
  2. *ff*($\mathsf{TA}_i.f$,$\mathsf{TA}_j.f$): as both TA_i and TA_j would finish together, there will be two possible time relations between their respective operation time intervals that are
     (a) *equals*($\mathsf{TA}_i[b_i,e_i]$,$\mathsf{TA}_j[b_j,e_j]$) where $b_i = b_j$ and $e_i = e_j$.
     (b) *finishes*($\mathsf{TA}_i[..,e_i]$,$\mathsf{TA}_j[..,e_j]$) where $e_i = e_j$.

- *Back-me* (Fig. 7): it has an initiator TA_i and a set of partners $0[\mathsf{TA}_j]n$ that the initiator asks when it fails to provision its functionality ① to the IoT ecosystem ②. *Back-me*'s characteristics include peer-to-peer collaboration between TA_i and $0[\mathsf{TA}_j]n$, common functionalities between all TAs ($\mathsf{TA}_i.f \cap 0[\mathsf{TA}_j.f]n \neq \phi$), and TAs synchronization to ensure that they are all active but in a mutually exclusive (either TA_i or the rest of TAs) way while the relation lasts. Although Figs. 6 and 7

Table 2 Scheduling techniques for IoT-ecosystem management

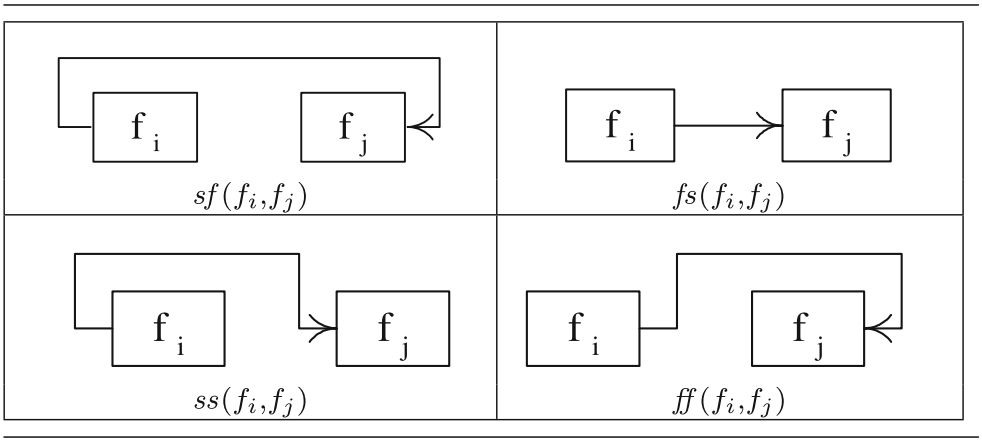


Table 3 Representation of relations in Allen's time algebra

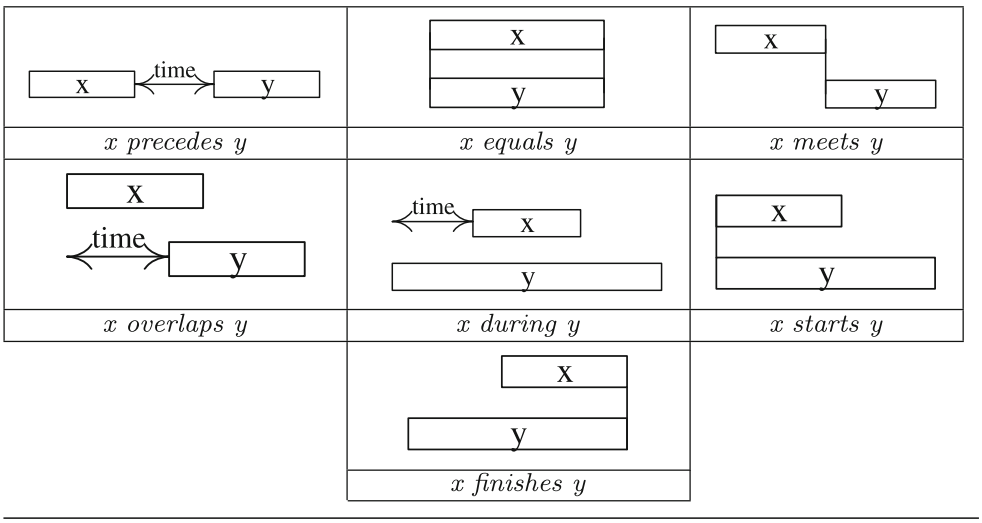


Fig. 5 Representation of *work-with-me*

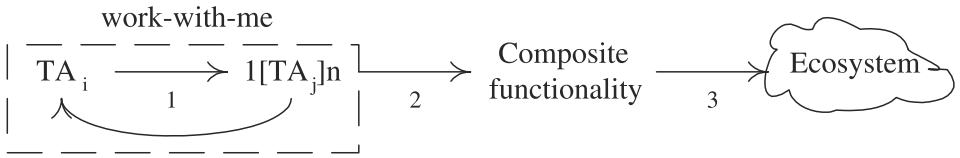
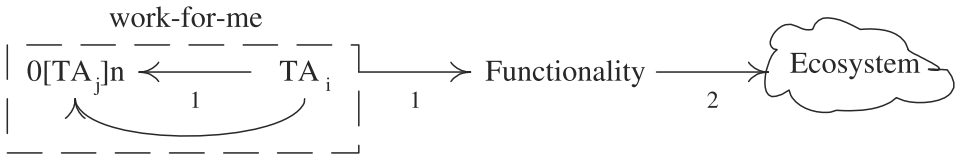


Fig. 6 Representation of *work-for-me*



look the same, there is a slight difference in the direction of the interaction connecting TA_i and $0[\mathrm{TA}_j]n$ together. In *work-for-me*, TA_i continues functioning after delegating its functionality to other TAs. Contrarily, in *back-me*, TA_i stops functioning requesting the assistance of TAs to complete its functionality. Let $\mathsf{TA}_{1,2,3}$ be medication dispenser, smart TV, and smart phone, respectively. We exemplify *back-me* between TA_2 and TA_3 with medicine reminders where the smart TV and medicine dispenser agree on displaying (i.e., *communicating*) the daily medicine intakes. However, due to the smart TV's unavailability, the smart phone acts as a substitute during the exchange with the medicine dispenser. Because *back-me* requires the active (but mutually exclusive) par-

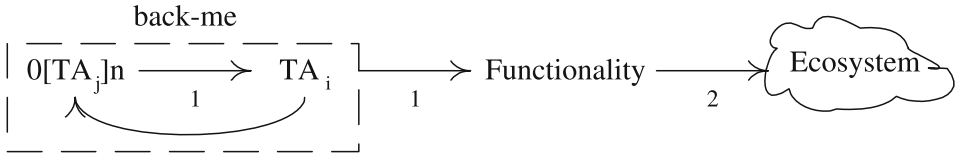


Fig. 7 Representation of *back-me*

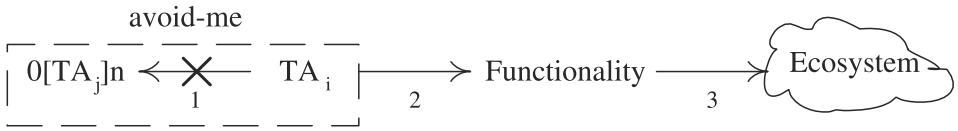


Fig. 8 Representation of *avoid-me*

ticipation of all TAs, synchronizing them would refer to specific scheduling techniques that, themselves, would refer to specific time relations between their respective operation time intervals as discussed below using TA_i and TA_j:

1. *fs*(TA_i,TA_j): as TA_j would start when TA_i finishes, there will be one possible time relation between their respective operation time intervals that is
   (a) *meets*(TA_i[..,e_i],TA_j[b_j,..]) where $e_i = b_j$.

- *Avoid-me* (Fig. 8): it has an initiator TA_i and a set of opponents 0[TA_j]*n* that are deemed undesirable to TA_i ① when provisioning its functionality ② to the IoT ecosystem ③. *Avoid-me*'s characteristics would include peer-to-peer opposition between TA_i and 0[TA_j]*n*, similar functionalities among all the TAs ($TA_i.f \cap TA_j.f = \phi$), and synchronization of all the TAs in a way that would prevent their simultaneous participation with other thing artifacts, TAs′, in other relations, whether *work-with-me*, *work-for-me*, or *back-me*, while *avoid-me* relation lasts. Let $TA_{1,2}$ be drug injection device and pill dispenser, respectively, and TA_1' be smart camera. We exemplify with *avoid-me* between TA_1 and TA_2 the recording of drug administration sessions preventing the simultaneous participation of TA_1 and TA_2 with TA_1' in *work-with-me* relation during these sessions, i.e., either *work-with-me*(TA_1,TA_1') or *work-with-me*(TA_2,TA_1').
  Because *avoid-me* requires the active (but sequential) participation of all TAs, synchronizing them would refer to specific scheduling techniques that, themselves, would refer to specific time relations between their respective operation time intervals as discussed below using TA_i and TA_j:

  1. *sf*(TA_i,TA_j): as TA_j would finish (for a while) when TA_i starts, there will be two possible temporal relations between their respective operation time intervals that are
     (a) *precedes*(TA_j[..,e_j],TA_i[b_i,..]) where $e_j - b_i < 0$.
     (b) *meets*(TA_j[..,e_j],TA_i[b_i,..]) where $e_j = b_i$.
  2. *fs*(TA_i,TA_j): as TA_j would start when TA_i has finished (for a while), there will be two possible temporal relations between their respective operation time intervals that are
     (a) *precedes*(TA_i[..,e_i],TA_j[b_j,..]) where $e_i - b_j < 0$.
     (b) *meets*(TA_i[..,e_i],TA_j[b_j,..]) where $e_i = b_j$.

4.4 Interaction flows between thing artifacts

Enacting *work-with-me*, *work-for-me*, *back-me*, *avoid-me* relations would support the on the fly formation of interaction flows between TAs participating in these relations as well as their life cycle synchronization. We recall that an interaction flow is a set of messages defined as a tuple ($< id, type, from/state, to/state, cnt >$) where $from/state$ and $to/state$ refer to the *sender* TA's and *receiver* TA's life cycles (Fig. 3), respectively. To identify the relevant messages per relation type, we draw some analogy with network protocols (e.g., [25]) resulting into *open*, *sync*, *ack*, and *close* messages. In Table 4, we consider 1 interaction flow per relation between TAs associated with 1 pairwise of scheduling technique and 1 Allen's time relation.

Let us consider an interaction flow linked to *work-with-me* between TA_i and TA_j. Acting as an initiator, TA_i takes on the prepared state in which it uses the *open* message to establish a communication channel with TA_j acting as a partner. Assuming that TA_j agrees on participating in the composite functionality that TA_i is putting together, TA_i and TA_j both transition from the prepared state to the activated state initiating their respective contributions to the completion of the composite functionality using the *sync* message. When TA_i ends the collaboration, TA_i uses the *ack* message to inform TA_j about its completion. Finally, the communication between them ends using the *close* message. By analogy with the discussion about *work-with-me*, let us consider *avoid-me*. The main feature here is that TA_i informs TA_j, acting as an opponent, about the enablement of a forthcoming relation with a peer requiring that TA_j either suspends or does not

Table 4 Messages between TA_i (initiator) and TA_j (participant)

Relation	Message *type*	*from/state*	*to/state*	*cnt*	Description
work-with-me	*open*	- TA_i. prepared	- TA_j.prepared	- TA_i. b_i & - TA_i. e_i	informs - TA_j about - TA_i's start- and end- times
- *ss* -	*ack*	- TA_j.prepared	- TA_i. prepared	null	confirms - TA_j readiness
- *equals* -	*sync*	- TA_i. activated	- TA_j.activated	$*[data]n$	initiates the concurrent execution of - TA_j & - TA_j
	ack	- TA_i.done	- TA_j.activated	success	confirm the completion of - TA_i
	close	- TA_i.done	- TA_j.done	null	ends the communication
work-for-me	*open*	- TA_i.prepared	- TA_j.prepared	- TA_i. b_i	informs - TA_j about - TA_i's start time
-*ss*-	*ack*	- TA_j.prepared	- TA_i.prepared	null	confirms - TA_j readiness
-*starts*-	*sync*	- TA_i.activated	- TA_j.activated	null	initiates the concurrent execution of - $\mathsf{TA}_i$and - TA_j
	ack	- TA_j.done	- TA_i.activated	success	confirms the completion of - TA_j
	close	- TA_i.activated	- TA_j.done	null	ends the communication
back-me	*open*	- TA_i.prepared	- TA_j.prepared	- TA_j	informs - TA_j about - TA_i's potential failure
- *fs* -	*ack*	- TA_j.prepared	- TA_i.prepared	null	confirms - TA_j readiness
-*meets*-	*sync*	- TA_i.failed	- TA_j.prepared	- TA_i. e_i	initiates the execution of - TA_j due to - TA_i's failure
	close	- TA_i.failed	- TA_j.activated	null	ends the communication
avoid-me	*open*	- TA_i.prepared	- TA_j.prepared or - TA_j.activated	null	informs - TA_j about its next partnership
- *sf* -	*ack*	- TA_j.suspended	- TA_i.prepared	null	confirms - TA_j's suspension
- *precedes* -	*close*	- TA_i.activated	- TA_j.suspended	null	ends the communication

initiate its operations. Upon TA_j's approval, TA_i proceeds with its relation while TA_j waits to be notified by TA_i, so that it either resumes or initiates its operations.

5 Impact of data quality on thing artifacts

As stated earlier, TAs participate in relations exchanging and acting on data. Depending on how "good" these data are, it may happen that a TA is involved in a conflicting situation with an opponent peer because of the data that it has received. In this section, we examine the impact of data quality on TAs. We list data quality concerns and then, how we address them.

5.1 Data quality cases

In [21], Qin et al. present an IoT-related data taxonomy categorized into data generation (specialized into velocity, scalability, dynamics, and heterogeneity), data quality (specialized into uncertainty, redundancy, ambiguity, and inconsistency), and data interoperability (specialized into semantics and incompleteness). Adopting Qin et al.'s second category as it falls into IoT ecosystems' data concerns, we focus on uncertainty that could be triggered because of missing readings by TAs and poor accuracy of reading outcomes by TAs, for example, redundancy that could be triggered because of duplicate reading outcomes from the same TA and similar reading outcomes from similar TAs nearby, for example, and, finally, inconsistency that could be triggered because of inappropriate reading outcomes by separate TAs, for example.

Case I:uncertainty is examined as per the following relations between TAs.

1. *work-with-me*: although some readings go missing from the main TA, the impacted TAs could continue functioning using the previous readings until the main TA addresses the missed readings. For illustration, when *work-with-me* connects the smart TV, remote control, and light switch together, suspending the smart TV will not affect the latter two; they can continue functioning based on previous settings until the smart TV resumes by sending fresh settings.
2. *work-for-me*: since some readings go missing from the main TA, this prevents the impacted TAs from receiving these readings. As a result, they suspend their operations due to absence of guidance from the main TA. We recall that in *work-for-me*, the impacted TAs act upon the request of the main TA. For illustration, if music streaming from the smart phone goes missing, the smart TV's operation will stand suspended until the music streaming resumes.
3. *back-me*: since some readings go missing from the main TA due to failure, the impacted TAs make up for the missing readings until the main TA is fixed by resuming the reading. For illustration, the smart TV's failure to display medication status will make the smart phone act as a substitute until the smart TV resumes working.
4. *avoid-me*: since some readings go missing from the main TA due to its failure, the opponent TAs make up for the missing readings with no option for reinstating the main TA. For illustration, the pill dispenser would make up for the lack of operation of the injection device.

Case II:redundancy includes 2 exclusive options because of duplicate readings from the main TA: (*i*) impacted TAs continue functioning because of idempotency or (*ii*) impacted TAs stop functioning. In either option, the TAs act differently because of the relation that they would have with the main TA. Let us begin with option (*i*).

- *work-with-me*: because of the duplicate readings from the main TA, the impacted TAs would end-up receiving the same outcomes twice, which means extra unnecessary work that would lead to the same outcomes, which could drain resources. For illustration, the remote control would be instructed to maintain the same brightness level that will be communicated again to the light switch.
- *work-for-me*: does not fit into redundancy and hence, is dropped from the discussion. The impacted TAs are expected to work for the main TA and do not address the redundancy concern.
- *back-me*: does not fit into redundancy and hence, is dropped from the discussion. The impacted TAs are expected to support the main TA's failure that could lead to lack of readings, but this is not the case.
- *avoid-me*: does not fit into redundancy and hence, is dropped from the discussion. The impacted TAs are activated only if the main TA is not activated, which is not the case.

We now discuss option (*ii*).

- *work-with-me*: on top of the risk of draining resources, the impacted TAs could end-up producing different readings since their operations are subject to the main TA's duplicate readings. This would require corrective measures to address the situation, as per our earlier work [14] and detailed in Sect. 5.2. For illustration, if the smart TV communicates the necessary adjustment of brightness to the light switch several times, the light switch would increase the brightness several times as well which could spoil the viewing experience.
- *work-for-me*, *back-me*, and *avoid-me* relations: are dropped for the same reasons in option (*i*).

Case III:inconsistency includes 2 exclusive options following sharing the main TA's inappropriate readings with the impacted TAs: (*i*) the TAs continue functioning since the readings are inappropriate and thus ignored or (*ii*) the TAs stop functioning because of inappropriate readings. Assuming that necessary mechanisms to identify and ignore inappropriate readings exist as described in [3], for example, option (*i*) is not discussed further. We focus, hereafter, on option (*ii*), only.

- *work-with-me*: inappropriate readings from the main TA would result in erroneous outcomes by the impacted TAs. This would call for corrective measures to address the situation, as per our earlier work [14] and detailed in Sect. 5.2. For illustration, if the smart TV communicates the wrong level of brightness to the light switch, this one will adjust this level in an inappropriate way.
- *work-for-me*: since the impacted TAs respond to the main TA's request, they do not expect any readings from the main TA. As a result, this relation is not applicable in this case.
- *back-me*: assuming that producing erroneous readings is also taken as a failure, the impacted TAs could be activated as a substitute until the main TA is fixed through corrective measures.
- *avoid-me*: does not apply here, since it does not have any means to activate the impacted TA in case the main TA is deemed to have failed. Here too, just like for *back-me*, we can assume that erroneous reading is also treated as a failure, thereby leading to the opponent TA to take over from the failed TA, as per the operation of *avoid-me* relation.

5.2 Adaptation to address data quality concerns

With the discussion above of data quality concerns, adaptation would target case II (redundancy) and case III (inconsistency), in particular, option (*ii*) *work-with-me*. Here, a main TA's failure will cause wrong execution of the impacted TAs. We need to apply corrective measures to these TAs. In our previous work [14], we associated 3 transactional properties with things, viz. compensatable, retriable, and pivot. If a thing is compensatable, its execution can be rolled back. For instance, a light switch can be rolled back to the right brightness. If a thing is retriable, its execution can be retried up to a prespecified number of times before failure is declared; alternatively, it could also be retried successfully before the maximum number of retries itself. If it is pivot, it is declared as a failure and reported as such.

Here too, we associate the same transactional properties with TAs where a property would be linked to the thing forming part of the TA in question. Hence, we say—via a slight abuse of notation—that each TA also has one of the 3 transactional properties above. Thus, if an impacted TA is compensatable, its execution is rolled back by implementing its life cycle in reverse back to the prepared state of Fig. 1. For illustration, if the light switch is showing the wrong brightness, it can be rolled back to the earlier brightness before the failure occurred. If the impacted TA is retriable, its execution is retried until successful execution or failure upon reaching the maximum number of allowed retries. For example, if the injection device does not work after the minimum number of retries, then it can be declared to have failed. These retries are attempted by instantiating new life cycles of the impacted TA in succession and rerunning them, until either one of them successfully completes, or all fail upon the maximum allowed number of retries. Finally, if it is pivot, it is declared as failed and reported as such.

6 Implementation

This section presents the system's architecture to implement an IoT ecosystems design, how this system was put into action, and finally how the experiments were carried out. For the moment, the system is restricted for internal use, only.

6.1 System's architecture

Figure 9 shows the architecture of the system demonstrating the technical feasibility of the concept of TA for IoT ecosystems design. The architecture refers to 4 modules (transformer, connector, deployer, and monitor) and 2 repositories (TAs-Relations and interaction flows).

It all starts when the IoT engineer defines things using for instance, the Web of Things (WoT) Thing Description (WoT-TD).[5] However, this does not fall into the scope of the current work. Next, the IoT engineer invokes the transformer that exposes the already defined things as TAs over dedicated WoT platforms like WoTify [10] or WoT Store [22]. Thing-TA coupling is one-to-many, i.e., a thing's functionality, whether primitive or composite, becomes a TA as per Sect. 4.2. After securing the necessary TAs, the connector establishes potential relations between these TAs based on their respective functionalities. For instance, *avoid-me* relation would refer to opponent TAs since they offer the same functionality. Once all the details, excluding operation time intervals, about the TAs and relations are stored in the TAs-Relations repository, the connector notifies the deployer about the readiness of TAs for deployment upon the request of end-users. To respond to these requests, the deployer parses the TAs-Relations repository looking for relevant TAs along with analyzing how they are related to each other as per Algorithm 1 whose outcome

[5] www.w3.org/TR/wot-thing-description.

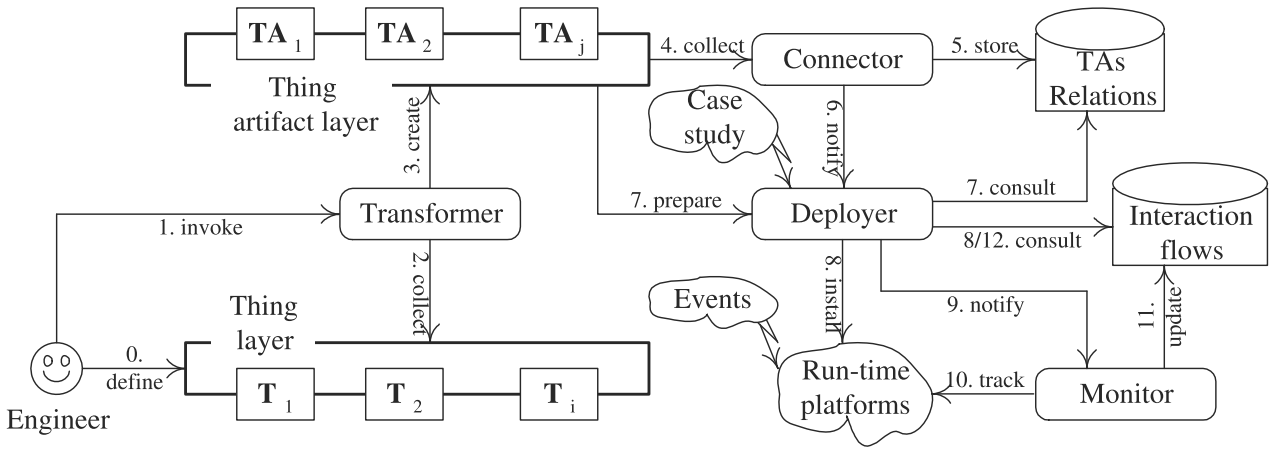


Fig. 9 System architecture for implementing TAs

is TAs orchestration obtained thanks to the scheduling techniques presented in Sect. 4.3. Although the selection of a particular scheduling technique could be at the IoT engineer's discretion, we recommend using data dependencies between TAs like presented in Algorithm 1. Once a scheduling technique is selected, the IoT engineer sets the values of the different TAs' operation time intervals. Based on these values, the necessary time interval relations between the TAs are identified as per Algorithm 2. These time interval relations are used to form interaction flows in compliance with Table 4.

After deploying TAs over run-time platforms, the deployer informs the monitor of the ongoing execution of these TAs. This one tracks the events that could impact this execution like TA failure and data inconsistency along with using relations to come up with solutions to these events. To this end, the monitor updates details about the interaction flows stored in the dedicated repository. In conjunction with the monitor's operations, the deployer consults the interaction flows repository to analyze the execution progress of TAs.

Algorithm 1: TA orchestration

Input: $\mathcal{TA}$ is the set of all TAs in the ecosystem; $\mathcal{TA}^{cs} \subset \mathcal{TA}$ is the set of all relevant TAs that would implement a case study (cs); $\mathcal{R}$ is the TAs-Relation repository
Output: σ is the orchestration of TAs included in $\mathcal{TA}^{cs}$

foreach $TA_i \in \mathcal{TA}^{cs}$ **do**
  Retrieve the set of all relations $(\mathcal{R}_i) \subset \mathcal{R}$ where TA_i act as an initiator;
  foreach $relation_{TA_i,TA_j} \in \mathcal{R}_i$ **do**
    switch $relation_{TA_i,TA_j}$ **do**
      case *work-with-me*
        if $(\mathsf{TA}_i.\{d^{in}\} \cap \mathsf{TA}_j.\{d^{in}\} \neq \emptyset)$
        then $\mathsf{schedule}_{\mathsf{TA}_i,\mathsf{TA}_j} \leftarrow \{\mathsf{ss}\}$;
        else $\mathsf{schedule}_{\mathsf{TA}_i,\mathsf{TA}_j} \leftarrow \{\mathsf{ff}\}$;
      case *work-for-me*
        if $(\mathsf{TA}_i.\{d^{in}\} \cap \mathsf{TA}_j.\{d^{in}\} \neq \emptyset$)
        then $\mathsf{schedule}_{\mathsf{TA}_i,\mathsf{TA}_j} \leftarrow \{\mathsf{ss}\}$;
        else $\mathsf{schedule}_{\mathsf{TA}_i,\mathsf{TA}_j} \leftarrow \{\mathsf{ff}\}$;
      case *back-me*
        $\mathsf{schedule}_{\mathsf{TA}_i,\mathsf{TA}_j} \leftarrow \{\mathsf{fs}\}$;
      case *avoid-me*
        $\mathsf{schedule}_{\mathsf{TA}_i,\mathsf{TA}_j} \leftarrow \{\mathsf{sf}\}$;
    Ask the IoT engineer to set values of operation time intervals as per $\mathsf{schedule}_{\mathsf{TA}_i,\mathsf{TA}_j}$;
    Add<relation $_{\mathsf{TA}_i,\mathsf{TA}_j}$, $\mathsf{schedule}_{\mathsf{TA}_i,\mathsf{TA}_j}$, AllenSel($\mathsf{TA}_i$,$\mathsf{TA}_j$) > to σ;
return σ;

6.2 System in operation

The system's simulator programmed in Python runs over configuration, execution, and monitoring stages. Each stage involves specific modules along with their interactions depicted in Fig. 9. During the configuration stage (involving the transformer and connector modules), the IoT engineer carries out the following steps:

1. **Run-time and Virtual Machines (VMs) configuration:** the engineer deploys run-time platforms as virtual machines, each labeled as a device in the simulator. These VMs have their own processing, storage, and communication capabilities. VMs also operate in their own isolated environment, including their own guest operating system, which is well suited for certain use cases in IoT simulations of multiple independent devices with varying characteristics.
2. **Organize devices into groups:** the devices are organized into groups, and each group is associated with a specific domain, such as CCTV and weather monitoring. The objective is to monitor the performance of each device and the TAs life cycles based on different settings/industries.

Algorithm 2: Allen relation selection (AllenSel)

Input: TA_i & TA_j
Output: $time_{\mathsf{TA}_i,\mathsf{TA}_j}$ denotes the identified time interval relation
2 **if** $(\mathsf{TA}_i.b_i = \mathsf{TA}_j.b_j)$ **then**
4 **if** $(\mathsf{TA}_i.e_i = \mathsf{TA}_j.e_j)$ **then** $time_{\mathsf{TA}_i,\mathsf{TA}_j} \leftarrow equals$;
6 **else** $time_{\mathsf{TA}_i,\mathsf{TA}_j} \leftarrow starts$;
7 **else**
9 **if** $(\mathsf{TA}_i.e_i = \mathsf{TA}_j.e_j)$ **then** $time_{\mathsf{TA}_i,\mathsf{TA}_j} \leftarrow finishes$;
11 **else**
12 **if** $(\mathsf{TA}_i.e_i > \mathsf{TA}_j.b_j \vee \mathsf{TA}_i.b_i < \mathsf{TA}_j.e_j)$ **then** $time_{\mathsf{TA}_i,\mathsf{TA}_j} \leftarrow precedes$;
13 **else**
14 **if** $(\mathsf{TA}_i.e_i = \mathsf{TA}_j.b_j \vee \mathsf{TA}_i.b_i = \mathsf{TA}_j.e_j)$ **then** $time_{\mathsf{TA}_i,\mathsf{TA}_j} \leftarrow meets$;
16 **return** $time_{\mathsf{TA}_i,\mathsf{TA}_j}$;

3. **Numbers of TAs:** engineer generates a random number (nb) of TAs per domain. The number of TAs is constrained by an interval of 2 values ($[min, max]$).
4. **TAs data:** according to TAs domain, engineer specify for each TA_k 2 sets of data, $(in_{k,1}, out_{k,2})$, related to the group to which TA_k belongs. These sets are $\{d_i^{in}\}$ and $\{d_j^{out}\}$ in Definition 4.1.
5. **TAs rate and relation probability:** this involve setting a probability of having relation (R) types between TAs occur, so that future generated TAs are assigned to these relation types as per their respective functionalities. Also, the TAs creation/generation rate is established, so that it determines how TAs-relations connections happen over time.
6. Finally, **Future-generated TAs:** sets expected start times for the future generated TAs based on their respective creation times.

In addition to the steps above, we associate a TA with a Processing Time (PT) defining how long it will lock a device, a deadline for getting a work done using this device once deployed, and an amount of data that it will process over this device. The deadline is relative to a TA's expected starting time and the amount of data binds a TA to a device according to its capacity.

In the execution stage (involving the deployer module), the IoT engineer initiates Algorithm 1, so that the simulator proceeds with every newly established TAs-relation connection obtained in the configuration stage's step 6 as follows:

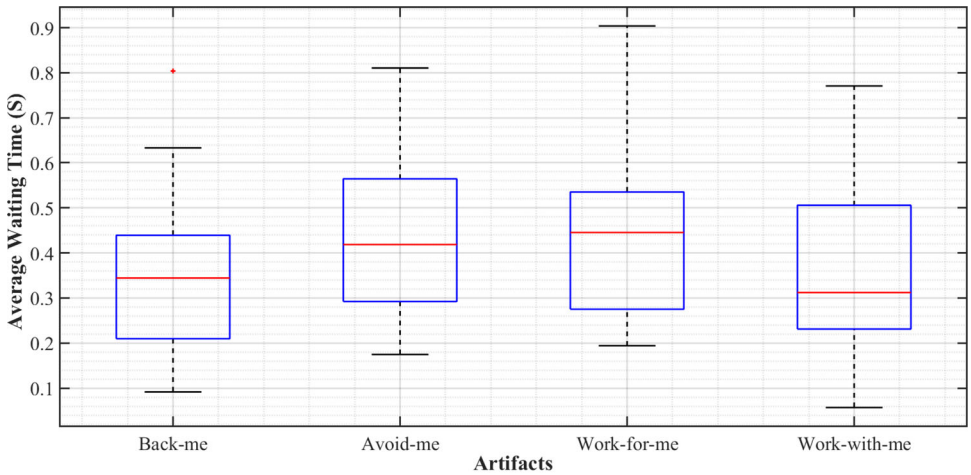
1. Initiate a particular scheduling technique based on the participating TAs' data as per Algorithm 1's lines 2-7.
2. Identify a time relation as per Algorithm 3 that corresponds to the selected scheduling technique in order to determine the TA that will start first.
3. Run the TAs participating in the identified time relation based on their respective start times over devices. It could happen that some devices do not have enough resources to host TAs. In this case, the deployment is put on-hold until some devices become available after other TAs either complete their works or are discarded because they do not participate in any relation.

In conjunction with the execution stage, the monitoring stage (involving the monitor module) makes the simulator "keep an eye" on the amount of available resources per device and track the execution progress of all TAs as per their respective life cycles (Definition 4.1). The simulator proceeds as follows.

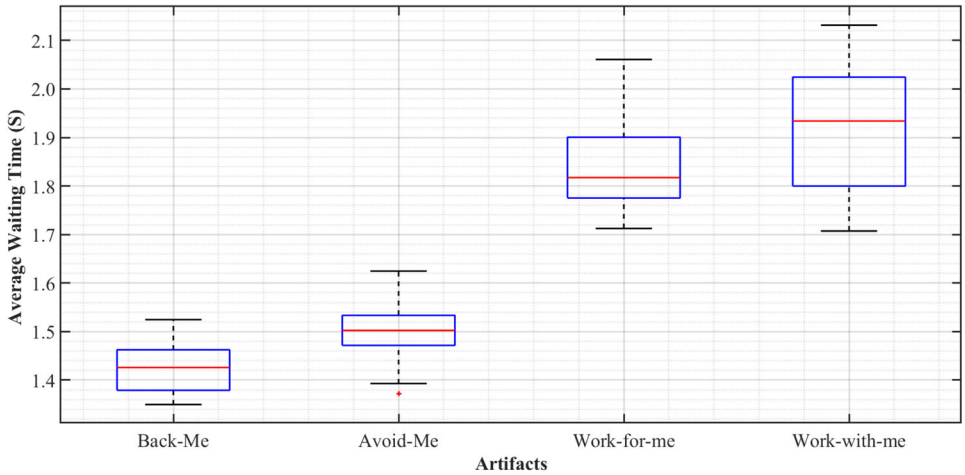
1. Track the events that could impact a device's load by checking whether
   (a) A TA execution was recently launched over this device (i.e., load increases).
   (b) A TA execution was either completed or rescheduled with the remaining time to complete (i.e., load decreases).
   (c) A TA execution was resumed (i.e., load increases).
2. Track the events that could impact TA execution by checking whether
   (a) A TA deployment was put on-hold (i.e., TA suspension).
   (b) A TA's deadline expired before the work is done (i.e., TA failure).

6.3 Experiments

The case study described in Sect. 3 forms the conceptual blueprint for the subsequent experiment's simulation of the IoT ecosystem with TAs. It sets the stage by outlining the design considerations, target TAs domain, and the data-centric approach, for example enhancing the quality of life for elderly people in a healthcare facility's living room. The simulation translates this theoretical case study into a practical and operational TAs interactive IoT system. It takes the predefined TAs and their relations and puts them in a simulated environment. In this way, the case study and the simulation are closely linked, with the former providing the conceptual scenario and the latter validating and refining its practical implementation to study how the number of relations per type impacts the response time of TAs. Given a certain R (set during the configuration stage's step 5), we carried out 2 experiments where the number of devices per domain was fixed to 10. In addition, we ran a scalability evaluation where we increased the number of TAs and number of devices per domain to further validate the feasibility of the work.



(a) $\mathsf{R}_i = 0.25$



(b) $\mathsf{R}_j = 0.7$

Fig. 10 Average waiting time of TAs to be executed per type of relation

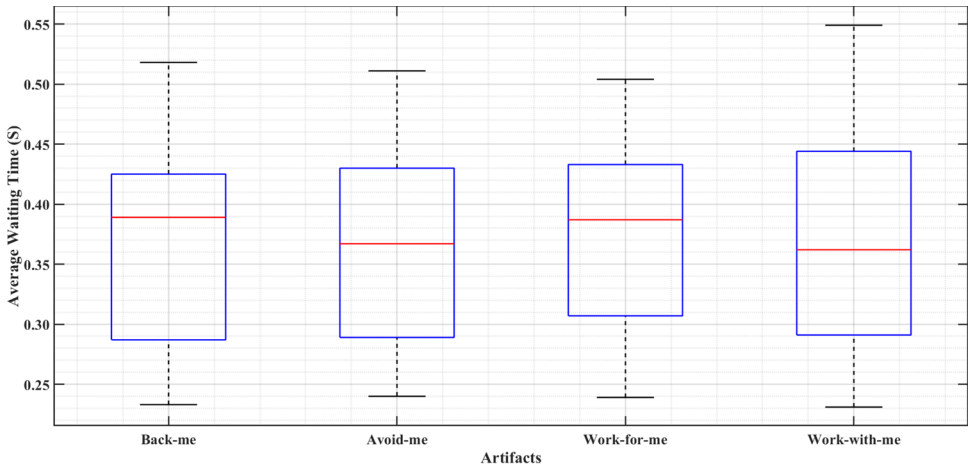
The first experiment examines the impact of R on how long TAs would wait before they are executed according to the relation type in which they participate. We considered 2 scenarios: (1) R is equally distributed among all the 4 types of relation (i.e., $\mathsf{R}_{i_{i=1,4}} = 0.25$), and (2) R is predominantly concentrated on 1 single type of relation and spreads equally over the rest (e.g., $\mathsf{R}_j = 0.7$ & $\mathsf{R}_{i_{i=1,4\ \&i\neq j}} = 0.1$). In both scenarios, the number (*nb*) of TAs generated per domain is randomly determined as per the configuration stage's step 3, and the processing time is the same for all TAs.

To ensure the experiment's representativeness, we iterated it 20 times and calculated statistics as per Fig. 10 boxplot. The results in Fig. 10 show the average waiting time of TAs per type of relation in each scenario. We observe that *avoid-me* and *back-me* relations provide the lowest waiting times compared to *work-with-me* and *work-for-me* relations when $\mathsf{R}_j = 0.7$ as per Fig. 10b. This can be explained by the fact that the former relations require one TA to be active at once while the latter relations impose more constraints (e.g., *ss* and *ff*) on TAs scheduling and this, regardless of R.

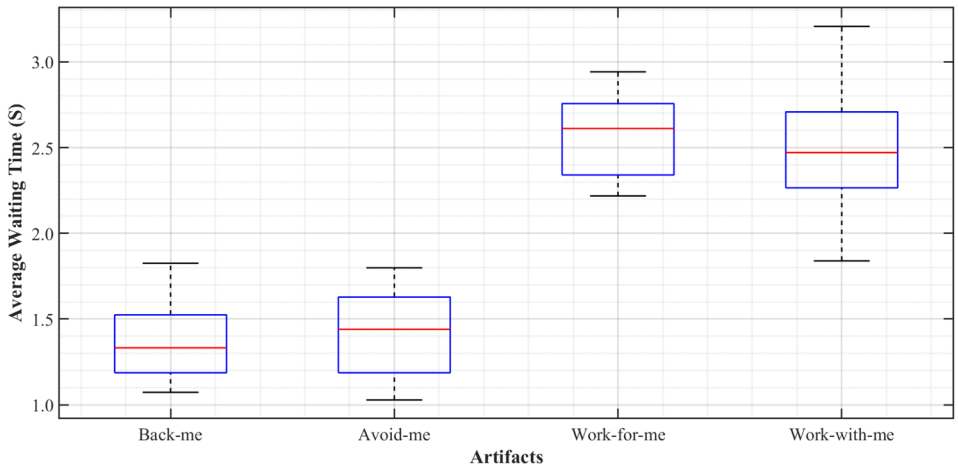
The second experiment that adopts the first experiment's settings assesses the impact of relation type on TAs' waiting times before execution as per the expected starting times. It could happen that a TA, whether initiator of or participant in a relation, waits longer because of a certain relation type and not because of the unavailability of a hosting device. For instance, *work-with-me* relation imposes that the initiator TA should wait for all the participating TAs to be active at the same time while *work-for-me* relation imposes that the participating TAs should be deployed but wait for the initiator TA. Table. 5 shows the distribution rate among all TAs that missed their expected starting times because of the waiting times. It is clear that when $\mathsf{R}_j = 0.25$ (TAs type of relation rate is almost the same) the relation type does not impact the waiting time. Contrarily, when $\mathsf{R}_j = 0.7$ (single dominate type of relation), the distribution rate confirms the few cases where the TAs missed their expected starting times because of the

Table 5 Rate of TAs having missed their expected starting times per type of relation

Rate	Work with Me	Work for Me	Back me	Avoid me
$R_i = 0.25$	26%	25%	26%	24%
$R_j = 0.7$	36%	31%	5%	28%



(a) $R_i = 0.25$



(b) $R_j = 0.7$

Fig. 11 Average waiting time of TAs to be executed per type of relation

relation type in which they participate. The *back-me* relation type has the lost missed starting time due to the nature of relation imposes on the participating TA to be deployed and then run upon the initiator TA's request

The scalability evaluation is to validate further the feasibility of the work. In the new experiments we ran the simulator with over 1K TAs and the number of devices per domain were set to 20 devices. The results obtained from the new configurations have shown a similar pattern to the previous results (Fig. 10) from previous experiments as per Fig. 11. Figure 11a shows the results when $R_{i_{i=1,4}} = 0.25$ (equal distribution among all the four types of relation), while in Fig. 11b $R_j = 0.7$ and $R_{i_{i=1,4\ \&i\neq j}} = 0.1$ (predominantly concentrated on one single type of relation and spreads equally over the rest). It is clear by increasing the number of TAs and having more iterations the performance remains consistent. We further extended this experiment to observe the waiting time per TA and whether the scheduling algorithm will scale linearly with the increment of the number of TAs. The result reported in Fig. 12 proves it scales linearly.

7 Conclusion

This paper presented thing artifacts as a novel concept that addresses the complexity of designing IoT ecosystems. A thing artifact is defined along 3 crosscutting aspects, namely functionality (what to perform?), life cycle (how to behave?), and interaction flow (with whom to exchange?). Each aspect sheds light on a specific perspective of an IoT ecosystem in

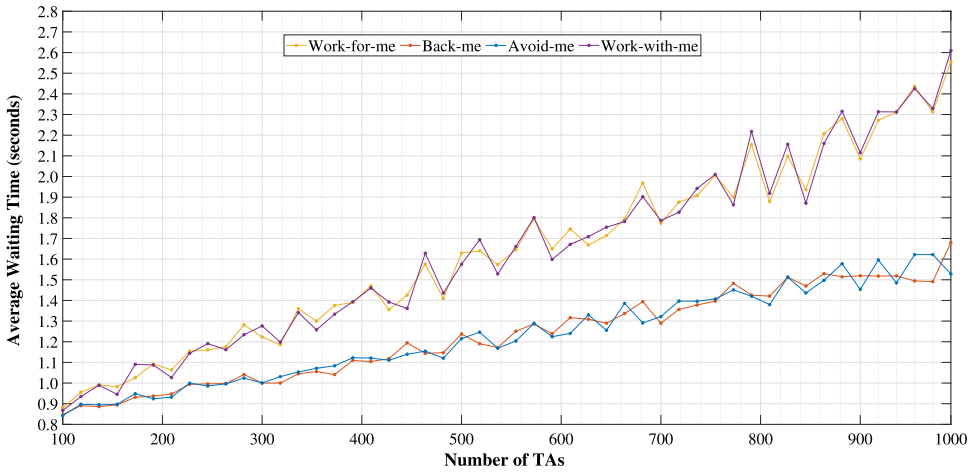


Fig. 12 Waiting time per number of TAs ($R_j = 0.7$)

terms of who are the residents of the ecosystem, what do they do, and how do they work together. Building upon these crosscutting aspects, thing artifacts engage in collaborative scenarios featuring 4 relations referred to as work-with-me, work-for-me, back-up, and avoid-me. Each relation has specific constituents to call upon and specific operational requirements to satisfy. To ensure successful collaborative scenarios, thing artifacts participating in these relations are synchronized using scheduling techniques that enforce their time availabilities at run-time. Along with the scheduling, data quality like uncertainty that could undermine the collaboration of thing artifacts is addressed through specific corrective measures. The paper also demonstrated the technical doability of thing artifacts through a set of experiments examining for instance, the impact of relation types on thing artifacts' waiting times before execution.

In terms of future work, on top of making the system available for the scientific community, we would like to examine the appropriateness of additional relations between thing artifacts. For instance, there are researchers in [4] who refer to some social-driven relations like parental, co-location, co-work, ownership, and social that could be relevant for the design of IoT ecosystems. We would also like to expand further the impact discussion of data quality on the "longevity" of IoT ecosystems. Uncertain, redundant, and inconsistent data could shorten things artifacts' "lifetimes." Finally, we would like to examine the role of relations between TAs in the dynamic management of IoT ecosystems. How to orchestrate TAs that could become involved in satisfying many users' demands? Could relations help prioritize their involvement? And, could relations help avoid conflicts on resources?

Acknowledgements The authors would like to thank Dr. Nanjangud C. Narendra from Ericsson Research in India for his contributions to the project.

Funding Open Access funding provided by the Qatar National Library.

Declarations

Conflict of interest The authors have no conflicts of interest to declare. All co-authors have seen and agree with the contents of the manuscript and there is no financial interest to report. We certify that the manuscript is original work and is not under review at any other publication.

References

1. Abdmeziem MR, Tandjaoui D, Romdhani I (2016) In A. Koubaa and E. Shakshuki, 2016editors, Robots and Sensor Clouds, chapter Architecting the Interne t of Things: State of the Art. Springer,
2. Allen JF (1983) Maintaining Knowledge about Temporal Intervals. Communications of the ACM, 26(11), November
3. Anh Nguyen T, Bucur D, Aiello M, Tei K (2013) Applying Time Series Analysis and Neighbourhood Voting in a Decentralised Approach for Fault Detection and Classification in WSNs. In Proceedings of the 4th International Symposium on Information and Communication Technology (SoICT'2013), Danang, Vietnam,
4. Atzori L, Iera A, Morabito G, 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),
5. Barnaghi PM, Sheth AP (2016) On Searching the Internet of Things: Requirements and Challenges. IEEE Intell Syst 31(6)

6. Bhattacharya K, Hull R, Su J (2009) A data-centric design methodology for business processes. Handbook for Research in Business Process Modeling, IGI Global
7. Cohn D, Hull R (2009) Business Artifacts: A Data-centric Approach to Modeling Business Operations and Processes. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 32(3), September
8. DZone (2017) The Internet of Things, Application, Protocols, and Best Practices. Technical report, https://dzone.com/guides/iot-applications-protocols-and-best-practices, (visited in May 2017)
9. Ghajargar M, Wiberg M, Stolterma E (2018) Designing IoT Systems that Support Reflective Thinking: A Relational Approach. International Journal of Design, 12(1),
10. Korkan E, Belhaj H, Hassine VE, Schlott, Käbisch S, Steinhorst S (2019) WoTify: A Platform to Bring Web of Things to your Devices, arXiv:1909.03296
11. Kumaran S, Liu R, Wu FY (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,
12. Maamar Z, Badr Y, Narendra NC (2010) Business Artifacts Discovery and Modeling. In Proceedings of the 8th International Conference on Service Oriented Computing (ICSOC'2010), San Francisco, California, USA
13. Maamar Z, Baker T, Sellami M, Asim M, Ugljanin E, Faci N (2018) Cloud *versus* Edge: Who Serves the Internet-of-Things Better? Internet Technology Letters, Wiley, 1(5)
14. Maamar Z, Sellami M, Narendra NC, Guidara I, Ugljanin E, Banihashemi B(2020) Towards an Approach for Validating the Internet-of-Transactional-Things. In Proceedings of the 34th International Conference on Advanced Information Networking and Applications (AINA'2020), Caserta, Italy
15. Miranda J, Mäkitalo N, García-Alonso J, Berrocal J, Mikkonen T, Canal C, Murillo JM (2015) From the Internet of Things to the Internet of People. IEEE Internet Comput 19(2):40–47
16. Narendra NC, Badr Y, Thiran P, Maamar Z (2009) Towards a Unified Approach for Business Process Modeling Using Context-Based Artifacts and Web Services. In Proceedings of the 2009 IEEE International Conference on Services Computing (SCC'2009), Bangalore, India
17. Nigam A, Caswell NS (2003) Business Artifacts: An Approach to Operational Specification. IBM Syst J 42(3):428–445
18. OpenKnowledge. Social Business Process Reengineering. Technical report, http://socialbusinessmanifesto.com/social-business-process-reengineering, 2012 (checked out in May 2016)
19. Popova V, Fahland D, Dumas M (2015) Artifact Lifecycle Discovery. Int J Coop Inf Syst 24(1)
20. Pticek M, Podobnik V, Jezic G (2016) Beyond the Internet of Things: The Social Networking of Machines. Int J Distrib Sensor Netw 12(6)
21. Qin Y, Sheng QZ, Falkner NJG, Dustdar S, Wang H, Vasilakos AV (2016) When Things Matter: A Data-Centric View of the Internet of Things. J Netw Comput Appl 64
22. Sciullo L, Aguzzi C, Felice M, Cinotti, TS (2019) WoT Store: Enabling Things and Applications Discovery for the W3C Web of Things. Las Vegas, USA, IEEE
23. Taivalsaari A, Mikkonen T (2017) A Roadmap to the Programmable World: Software Challenges in the IoT Era. IEEE Softw 34(1)
24. Weske M (2012) Business process management architectures. Springer, In Business Process Management
25. Wondracek G, Comparetti PM, Kruegel C, Kirda E (2008)Automatic Network Protocol Analysis. In Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS'2008), San Diego, CA, USA
26. Zorzi M, Gluhak A, Lange S, Bassi A (2010) From Today's INTRAnet of Things to a Future INTERnet of Things: a Wireless- and Mobility-related View. IEEE Wirel Commun 17(6)