Towards an Approach for Validating the Internet-of-Transactional-Things

Zakariac Maamar[1], Mohamed Sellami[2], Nanjangud C. Narendra[3], Ikbel Guidara[4], Emir Ugljanin[5], and Bita Banihashemi[6]

Abstract This paper examines the impact of transactional properties, known as pivot, retriable, and compensatable, on Internet-of-Things (IoT). Despite the ever-growing number of things in today's cyber-physical world, a limited number of studies examine this impact while considering things' particularities in terms of reduced size, restricted connectivity, continuous mobility, limited energy, and constrained storage. To address this gap, this paper proceeds first, with exposing things' duties, namely sensing, actuating, and communicating. Then, it examines the appropriateness of each transactional property for each duty. During the performance of transactional things, (semi)-atomicity criterion is adopted allowing to approve when these things' duties could be either canceled or compensated. A system that runs a set of what-if experiments is presented in the paper allowing to demonstrate the technical doability of transactional things.

1 Introduction

To ensure that Internet-of-Things (IoT) becomes the technology of choice when designing and developing *mission-critical applications* like air-traffic control and nuclear-plant monitoring, ICT professionals should examine things from a reliability perspective. Unfortunately, this is barely happening!! Our literature review resulted into a very limited number of works on IoT reliability. In the context of software, reliability is "*the probability of failure-free software operation for a specified period of time in a specified environment*" [3]. To address the research gap in IoT reliability, we adopt transactional properties that are specialized into *pivot*, *retriable*, and *compensatable* [15]. Thanks to these properties, application engineers can define the "acceptable" execution behaviors of IoT applications. By acceptable

[1]Zayed University, UAE · [2]Samovar, Télécom SudParis, Institut Polytechnique de Paris, France. · [3]Ericsson Research, India · [4]Lyon 1 University, France · [5]State University of Novi Pazar, Serbia · [6]York University, Canada.

we mean when a failed execution of an application is tolerated and when it is not, for example. Transactional properties are commonly used in many ICT domains; cloud computing [5], databases [11], Web services [18], and business processes [16].

ICT professionals' limited attention to IoT reliability is, in fact, shadowed by other pressing concerns that are "forcing" things to act independently from each other, i.e., silos [19]. These concerns include diversity and multiplicity of things' development and communication technologies [2], users' reluctance and sometimes rejection because of privacy invasion that things cause [4], limited IoT-platform interoperability [7], lack of killer applications that justify the existence of things [13], lack of an IoT-oriented software engineering discipline that would guide thing analysis, design, and development [8], and, finally, passive nature of things that primarily act as data suppliers (with some actuating capabilities) [20].

According to Gartner (`www.gartner.com/newsroom/id/3165317`), 6.4 billion connected things were in use in 2016, up 3% from 2015, and will reach 20.8 billion by 2020. This large and ever-growing number of things need to be "harnessed" so that things would be considered when designing and developing mission-critical applications. Our work includes 2 steps. In the first step, we shed light on duties of things that are *sensing*, *actuating*, and *communicating*. In the second step, we examine the appropriateness of each transactional property, namely *pivot*, *compensatable*, and *retriable*, for each duty. We raise many questions like, is pivot appropriate for sensing? and if yes, how does pivot impact sensing? Our appropriateness analysis allows to devise transactional things that would expose specific transactional properties. This exposure depends on requirements imposed on things' duties such as re-executing a failed duty **must** continue **until** successful execution and compensating a successful duty **must-not** happen **after** execution. For each situation, a transactional property needs to be selected.

Our contribution is manifold: (*i*) definition of things' duties, (*ii*) application of transactional properties to things' duties, whether these duties are atomic or composite, (*iii*) adoption of (semi-)atomicity criterion during the analysis of transactional things that offer composite duties, and (*iv*) proof-of-concept system demonstrating the technical doability of transactional things. Section 2 defines transactional properties, presents some related work, and then introduces a case study. Section 3 discusses transactional things in terms of duties and coupling transactional properties to things. The system demonstrating transactional things is presented in Section 4. Concluding remarks and future work are both reported in Section 5.

2 Preliminaries

2.1 Definitions

Transactional properties are primarily used for "declaring" the (un)acceptable execution behaviors of a computational unit in term of either success or failure [15].

The nature of this unit varies from one ICT domain to another. In the context of business processes, the unit is task (used here for illustration) and in the context of databases, the unit is transaction. In a business process a task is pivot if once it successfully completes, its execution effects remain unchanged forever and cannot be undone. Additionally, a pivot task cannot be retried following failure. A task is compensatable if its effects after successful execution can be semantically undone. And, a task is retriable if it is sure to successfully complete after several finite activations.

2.2 Related work

Despite the common use of transactions in many ICT domains, transactional things are overlooked except of the work of Vidyasankar [21, 22]. In the following, we briefly discuss transactional Web services as an illustrative domain. In [14], Li et al. derive transactional properties of composite Web services based on the properties of their component Web services. Li et al. resort to some workflow constructs such as sequence, concurrency, alternative, and iteration to define the transactional properties of the workflow underpinning the composite Web service and to determine whether this workflow is schedulable or not based on the transactional properties of its different fragments. In [9], Ding et al. examine transactional properties of component and composite Web services to evaluate the performance of basic workflow patterns such as sequence, parallel, and selection. In [10], a transaction-based Web services composition model is proposed with focus on composite Web services that satisfy specific atomicity constraints. In fact, a Web service can be either atomic, semi-atomic, or non-atomic. Because of the atomicity constraint over the composite Web service, only specific Web services are allowed to participate in this composite Web service. The proposed model exploits the transactional properties of Web services during composition to fulfill constraints and preferences of the designer and/or user of the composite Web service.

A very limited number of works consider transactional properties in IoT. In [21], Vidyasankar defines these properties of IoT service (and **not IoT**, only) compositions namely atomicity, retriability, compensatability, and pivot to verify IoT service execution. To handle the variety of IoT applications compared to conventional Web services, Vidyasankar suggests triggering and continuous execution. They are used to refine the definition of transactional properties to use in IoT. Vidyasankar also proposes in [22] a transaction model for the execution of IoT service compositions. He relaxes the atomicity and isolation properties for IoT service compositions by preserving composition and batch order, restricting atomicity to parts of the transaction, and executing transactions, only, partially. Although Vidyasankar's work targets transactional things, it remains at a high level of granularity by focusing on services of things and treating things as a "**sealed**" unit that could have transactional properties. Our work is at a low level of granularity by focusing on (atomic and composite) duties of things and how transactional properties impact these duties.

2.3 Case study

We assume a smart city that runs many applications (app) such as Transportation ($\mathcal{T}_{app}$) for traffic control and Environment ($\mathcal{E}_{app}$) for air-pollution monitoring. Our case study is a temporarily-closed tunnel that results from a car accident. The accident is primarily reported to $\mathcal{T}_{app}$ by the tunnel's cameras and, also, the tweets of some fellow drivers. To inform other drivers about the accident, $\mathcal{T}_{app}$ uses different communication means (e.g., variable-message signs, flip-disc display systems, and highway-advisory radio stations) that are installed across the city and on the borders of highways. As a result of the accident, the carbon-dioxide level inside the tunnel dramatically increases; some fans did not work or stopped working, forcing $\mathcal{E}_{app}$ to issue warnings to emergency teams.

From a reliability perspective, we identify some cases that show the importance of applying transactional properties to certain things' execution behaviors (e.g., failure is accepted and success can be reversed):

1. Sensors (s_i) streaming air quality data to $\mathcal{E}_{app}$ and cameras (c_j) streaming live content to $\mathcal{T}_{app}$ are deemed retriable. Because car accidents could result into destroying some equipment, it is a must to guarantee successful streaming through multiple retrials. The same applies to fans (f_k); being retriable would guarantee a better air circulation in the tunnel.
2. Barrier (b_l) closing the tunnel will be opened again upon the authorities' approval to resume the traffic. Thus, the barrier is deemed compensatable. Should the barrier fail to close, a traffic cone is used to block the access making the barrier's temporary failure acceptable.
3. Flip-disc display system (fd_m) is responsible for informing drivers about the accident. This is done in collaboration with other variable-message signs and highway-advisory radio stations. As a result, the flip-disc display system's failure does not constitute a major concern and, thus, is deemed pivot.

3 Approach for coupling transactional properties to things

3.1 Duties of things

In [17], we identified 3 duties of things that are *sensing* (collecting/capturing data, e.g., fans), *actuating* (processing/acting upon data, e.g., barriers), and *communicating* (sharing/distributing data, e.g., flip-disc display). These duties are either enabled or disabled ((0,1) in Fig. 1) according to the requirements of the under-development IoT applications (e.g., $\mathcal{T}_{app}$ and $\mathcal{E}_{app}$). From a duty perspective, a thing senses the cyber-physical surrounding so that it generates data; a thing actuates data including those that are sensed; and a thing communicates with the cyber-physical surrounding the data that is sensed and/or actuated. Receiving data and/or commands from external parties (e.g., other things) is also taken care by the communicating duty.

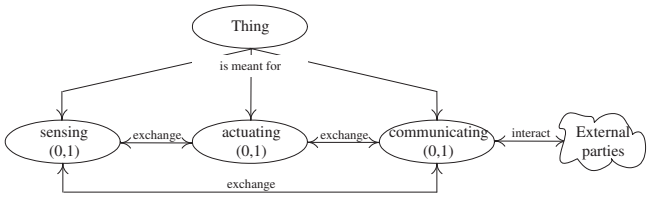

Fig. 1 Duties associated with a thing

Thanks to our duty classification, we can compose duties as per the following 4 representative cases (other cases like Communicating → Actuating and Communicating → Actuating → Sensing are not listed):

1. Sensing→Actuating→Communicating: sensed data are passed on to actuating; and the data that result from actuating are passed on to communicating for distribution. An example is when a microcontroller fitted into a barrier assesses the level of carbon-dioxide (sensing duty) before triggering this barrier closure (actuating duty) and informing the traffic control center (communicating duty) of the closing.
2. Sensing→Actuating: sensed data are passed on to actuating; and the data that results from actuating is final. An example is when a microcontroller fitted into a barrier assesses the level of carbon-dioxide (sensing duty) before triggering this barrier closure (actuating duty).
3. Sensing→Communicating: sensed data are passed on to communicating for distribution. An example is when sensors submit their data to the traffic control center for archive.
4. Actuating→Communicating: data that result from actuating are passed on to communicating for distribution to other parties.

3.2 Overview of the approach

Fig. 2 illustrates the general representation of our approach for coupling transactional properties to things. It includes 3 modules, selection, coupling, and validation, 1 repository of things along with their duties, and 2 lists containing respectively transactional properties and validation rules (Tables 1, 2 and 3). The IoT engineer uses the selection module to identify the component/composite things and their atomic/compsite duties that will be included in the under-development IoT application. Once selected, the things/duties are associated with specific transactional properties thanks to the coupling module. Prior to deploying the transactional things/duties, this coupling is subject to validation to ensure the appropriateness of these transactional properties and hence, satisfaction of the IoT engineer's needs. Indeed, the failure of a thing like barrier remains acceptable as long as a substi-

tute to the failed barrier exists. Coupling and validation are examined in Section 3.3 and 3.4 and implemented in Section 4.

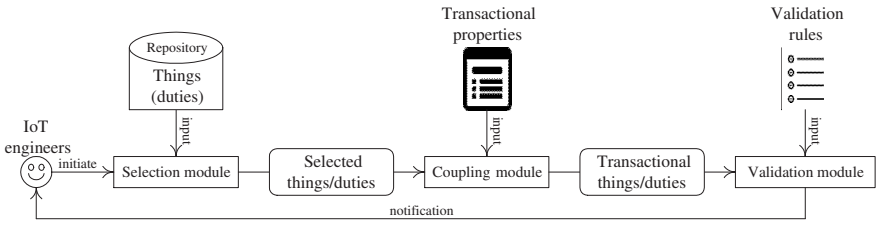

Fig. 2 Approach for coupling transactional properties to things

3.3 Coupling transactional properties to component things

To couple transactional properties to component (i.e., taken separately) things, we examine **the appropriateness (i.e., is it applicable or not?) of each property for a thing's particular duty**. We consider atomic duties and then, composite duties. In either case, the duties are related to the same thing.

3.3.1 Atomic duties

1. Sensing duty: Pivot coupled to sensing should not arise; a failed sensing would deprive the IoT application from sensed data; Retriable coupled to sensing could arise to guarantee sensed data availability to the IoT application; and Compensatable coupled to sensing should not arise; a failed sensing is not acceptable (like pivot) and canceling (in the sense of recalling/undoing) what has been sensed is not doable, too.
2. Actuating duty: Pivot coupled to actuating should not arise; a failed actuating would deprive the IoT application from actuated data; Retriable coupled to actuating could arise first, to guarantee actuated data availability to the IoT application and/or second, to allow the IoT application to actuate data further, if necessary; and Compensatable coupled to actuating could arise; a failed actuating is acceptable and canceling (in the sense of reversing) what has been actuated is doable.
3. Communicating duty: Pivot coupled to communicating should not arise; a failed communicating would deprive the IoT application from interacting with external parties; Retriable coupled to communicating could arise to guarantee data availability to the IoT application and external parties; and Compensatable coupled to

communicating should not arise; a failed communicating is not acceptable (like pivot) and canceling (in the sense of recalling/undoing) what has been communicated is not doable.

Table 1 summarizes the appropriateness of pivot, retriable, and compensatable for every atomic duty in a thing; no/yes refers to not-applicable/applicable.

Table 1 Summary of transactional properties/atomic duties coupling

Transactional property	Thing's atomic duties		
	Sensing	Actuating	Communicating
Pivot	no	no	no
Retriable	yes	yes	yes
Compensatable	no	yes	no

3.3.2 Composite duties

To decide on the appropriateness of a transactional property for a particular composite duty, we resort to the concept of (full) atomicity [21]. Atomicity declares the success of a particular combination of duties when **all** duties in this combination succeed. Otherwise, the combination fails requiring the (successful) compensation of those duties that succeeded.

1. Sensing→Actuating→Communicating duties: Pivot coupled to sensing then actuating then communicating should not arise because of the risk of not achieving the atomicity of this combination. Indeed, the atomicity is achieved, only, if all the 3 duties are successfully achieved, which might not happen since some duties could fail; Retriable coupled to sensing then actuating then communicating could arise because of the guarantee of successfully executing all the 3 duties, which achieves the atomicity of this combination; and Compensatable coupled to sensing then actuating then communicating should not arise like pivot.
2. Sensing→Actuating duties: Pivot coupled to sensing then actuating should not arise because of the risk of not achieving the atomicity of this combination. Indeed, this atomicity is achieved, only, if the 2 duties are successfully achieved; Retriable coupled to sensing then actuating could arise because of the guarantee of successfully executing the 2 duties, which achieves the atomicity; and Compensatable coupled to sensing then actuating should not arise like pivot.
3. Sensing→Communicating duties: Pivot/Compensatable coupled to sensing then communicating should not arise since both transactional properties do not apply to sensing and communicating (Table 1); Retriable coupled to sensing then communicating could arise because of the guarantee of successfully executing the 2 duties, which achieves the atomicity.
4. Actuating→Communicating duties: Pivot/Compensatable coupled to actuating then communicating does not arise since both transactional properties do not

apply to communicating (Table 1) and Retriable coupled to actuating then communicating could arise because of the guarantee of successfully executing the 2 duties, which achieves the atomicity.

Table 2 summarizes the appropriateness of pivot, retriable, and compensatable for different composite duties in a thing. In this Table 1 and 0 refer to execution success and failure, respectively (e.g., sac: 110 means execution success for sa and execution failure for c).

Table 2 Summary of transactional properties/composite duties coupling: the case of atomicity

Transactionl property	Composite duty execution outcome	Atomicity
Pivot/ Compensatable	sac: 111	yes achieved
	sac: 110	not achieved with both sensing and actuating requiring compensation
	sac: 10-	not achieved with only sensing requiring compensation
	sac: 0–	not applicable
	sa : 11	yes achieved
	sa : 10	not achieved with only sensing requiring compensation
	sa : 0-	not applicable
Retriable	sac: 111	yes achieved
	sa : 11	yes achieved
	sc : 11	yes achieved
	ac : 11	yes achieved

3.4 Coupling transactional properties to composite things

A composite thing is built-upon a set of component things that are connected through dependencies such as *finish-to-start*, *start-to-start*, *finish-to-finish*, and *start-to-finish*. We analyze *finish-to-start* since it is commonly used in composition scenarios. For the sake of illustration, we assume that directly dependent things offer exclusive duties except of the communicating duty that allows interactions between things to happen. Example of the assumption is that when a thing (t_i) senses, the dependent thing (t_{i+1}) actuates, only; it is represented as $\mathsf{s}_{\mathsf{t}_i} \rightarrow \mathsf{c}_{\mathsf{t}_i} \rightarrow \mathsf{c}_{\mathsf{t}_{i+1}} \rightarrow \mathsf{a}_{\mathsf{t}_{i+1}}$ where s, c, and a stand for sensing, communicating, and actuating, respectively.

In preparation for coupling transactional properties to composite things, we list in Table 3 the possible duties that dependent component things are allowed to perform when they participate in the same composite thing and in compliance with our duty-exclusivity assumption.

By analogy to Web services, full atomicity is a key element to the execution of composite Web services; it guarantees that this execution concludes in a consistent state even in the presence of failure [6]. However, in the context of IoT full atomicity is not applicable. Composite things are expected to operate on continuous streams

Table 3 Possible duties of dependent component things in a composite thing

t_i's duties	t_{i+1}'s duty	Description/Example
sc	ca	t_{i+1} performs an actuation upon receiving sensed data from t_i; e.g., detection of increased carbon dioxide levels in the tunnel results in the barrier being opened
	cac	*in addition to the above*, t_{i+1} also communicates the results of its actuation to a third party
sac	c	t_{i+1} communicates to a third party the result of actuating the sensed data received from t_i
c	csac	t_{i+1} performs sensing based on the received communication, then performs an actuation based on the sensed data, and finally, communicates the results of this actuation to a third party; e.g., when the fans get switched off in the tunnel, the air quality streaming data is communicated, and based on the air quality, the barriers are opened.
	cac	t_{i+1} performs an actuation based on the received communication and then communicates the outcome of this actuation to a third party.
	ca	t_{i+1} performs an actuation based on the received communication.
	cs	t_{i+1} performs sensing based on the received communication.
	csc	t_{i+1} performs sensing based on the received communication and then communicates what has been sensed to a third party.
ac	csc	t_{i+1} performs additional sensing after communicating the result of the previous actuation, and also communicates the sensed data; for e.g., after the barriers are opened, and this is communicated, the air quality sensors could once again sense the air quality and communicate the (hopefully improved) air quality.
	cs	t_{i+1} senses the actuation result.
	c	t_{i+1} communicates the result of actuation to a third party.

of data [12]. As a result, we leverage the concept of semi-atomicity as introduced in [1][1], i.e., a relaxed notion of full atomicity. This also happens to be feasible in the case of executing composite things since the composite duties of component things - sa, sc, ac, and sac - are either retriable or compensatable - please recall from the discussion about composite duties that pivot is not allowed for these composite duties. Therefore, we represent a composite thing's sequence of duties, without loss of generality, as alternating sequences of compensatable and retriable duties. A failure can impact either a compensatable duty or a retriable duty (when constrained by a maximum number of retrials), leading to the following possibilities:

1. Sequence of compensatable duties followed by an atomic retriable duty, which fails: full atomicity is achieved here requiring to successfully compensate the sequence of duties after the failure of the retriable duty (maximum number of retrials reached).
2. Sequence of compensatable duties followed by a sequence of more than one retriable duty, of which the latest one fails: semi-atomicity is achieved here, since the previous retriable duties have all been successful.

[1] Semi-atomicity, a weaker form of atomicity, allows a global transaction to commit even it some subtransactions abort, provided that their alternative subtransactions commit [23].

3. Sequence of retriable duties followed by a single compensatable duty, which fails: semi-atomicity is achieved here due to the failure of the compensatable duty.
4. Sequence of retriable duties followed by a sequence of more than one compensatable duty, of which the latest one fails: similar to the previous case, i.e., here too, semi-atomicity is achieved.
5. Sequence of compensatable duties, of which the latest one fails: semi-atomicity is achieved subject to successfully canceling the completed duties.
6. Sequence of retriable duties, of which the latest one fails: semi-atomicity is achieved here.

4 Implementation

We developed an online system (`transactional.connect.rs/`) for controlling things and assigning duties and transactional properties to them. 3 What-If-Analysis scenarios have been implemented as per the following details.

1. What-If-Analysis$_1$(component thing, atomic duty): the user assigns one duty at a time to a thing, and one transactional property to that duty. Based on the user's selection, the system displays whether that assignment is approved or not (Table 1).
2. What-If-Analysis$_2$(component thing, composite duties): the user assigns many duties to a thing, and one transactional property to each duty. Based on the user's selection, the system displays whether atomicity is achieved or not (Table 2).
3. What-If-Analysis$_3$(composite things, composite duties): like the other 2 cases, the user assigns one duty and one property to each thing in the composition. Based on the user's selection, the system displays when either atomicity or semi-atomicity is achieved.

In all the cases, the user can configure the things by switching them on and off, changing their initial parameters, and checking their current status. We simulated things by writing several Python programs running on a Linux server. However, the data produced is "real". The dashboard synchronizes with these programs ensuring real-time display of all necessary parameters. Thing communication uses MQTT protocol and the communication between the dashboard and things uses websocket MQTT Paho library that runs in the browser by JavaScript. Things' controlling instructions chosen on the dashboard are sent via MQTT, and all things subscribe to those instructions. We built the dashboard using PHP, MySQL, JavaScript, HTML, and CSS and deployed it on a Linux server. The things involved in the implementation are as follows:

1. Outside temperature sensor provides temperature in $^{\circ}$C.
2. Tunnel ambient temperature sensor measures the current temperature in the tunnel and could be switched on and off. This temperature depends on the ventilator rotating speed and outside temperature and is calculated using Equation 1:

$$Temp_{tunelTemp} = Temp_{outsideTemp} - (Speed_{ventilator}/1000) \quad (1)$$

3. Ventilator actuator has a maximum rotating speed of 12000RPM (Revolutions Per Minute). Its purpose is to adjust the tunnel's ambient temperature by increasing or decreasing the speed. A 100RPM increase results in lowering the ambient temperature by $0.1°\mathrm{C}$ and *vice-versa*. Having the ventilator work at a maximum speed of 12000RPM makes it fail. The dashboard allows us to switch the ventilator on and off and/or to increase or decrease the rotating speed.
4. Automatic barrier actuator is opened when no issues are reported and *vice-versa*. Examples of issues include tunnel's ambient temperature going over $30°\mathrm{C}$, not receiving temperature measurements, and not receiving information from the ventilator. The barrier is subscribed to both temperature and ventilator data so that decision of closing or opening the barrier is made.
5. Variable-message sign sensor that does not influence anything, but shows notification based on the barrier state, and can be switched on and off.

What-If-Analysis$_3$ implements our case study and allows to check whether the atomicity of the system is achieved. It is when the tunnel's ambient sensor is *retriable*, the ventilator is *retriable* or *compensatable*, the automatic barrier is *retriable* or *compensatable*, and the variable message sign is *retriable*.

5 Conclusion

A successful integration of IoT-compliant things into mission-critical applications requires a transactional analysis of these things. In this paper we analyzed the coupling of transactional properties (pivot, compensatable, and retriable) to single and composite things. We defined things' 3 duties (sensing, actuating, and communicating) and then applied the transactional properties to these duties. This allowed us to declare when either the successful execution of a thing's atomic duties or composite duties is compensated or the failed execution of a thing's atomic duties or composite duties is accepted. Along with this declaration, we considered (full/semi)-atomicity as an additional criterion to approve the compensation of successful executions. A set of what-if experiments have been carried out using the tunnel closure case-study to illustrate the different recommendations put in place.

In term of future work we would like first, to consider other forms of dependencies (e.g., *start-to-start*) between things in compositions and second, to adopt model checking for verifying the compliance of the operational behavior (i.e., process model) of a given thing (or a collection of things) with the specific transactional properties of this thing (or this collection of things). We also plan to investigate developing tools that would automate the process of checking this compliance.

References

1. Aidong, Z., Marian, N., Bharat, B., Omran, B.: Ensuring Relaxed Atomicity for Flexible Transactions in Multidatabase systems, vol. 23 (1994)
2. Androcec, D., Tomaš, B., Kišasondi, T.: Interoperability and Lightweight Security for Simple IoT Devices. In: ISS'2017. Opatija, Croatia (May 2017)
3. ANSI/IEEE: Standard Glossary of Software Engineering Terminology, STD-729-1991, ANSI/IEEE (1991)
4. Barhamgi, M., Perera, C., Ghedira, C., Benslimane, D.: User-centric Privacy Engineering for the Internet of Things. IEEE Cloud Computing **5**(5) (2018)
5. Bellaaj, F., Brabra, H., Sellami, M., Bhiri, S., Gaaloul, W.: A Transactional Approach for Reliable Elastic Cloud Resources. In: SCC'2019. Milan, Italy (2019)
6. Bhiri, S., Perrin, O., Godart, C.: Ensuring required failure atomicity of composite Web services. In: WWW'2005. Chiba, Japan (2005)
7. Bröring, A., Ziller, A., Charpenay, V., Schmid, S., Thuluva, A., Anicic, D., Zappa, A., Linares, M., Mikkelsen, L., Seidel, C.: The BIG IoT API - Semantically Enabling IoT Interoperability. IEEE Pervasive Computing **17**(4) (August 2018)
8. Dastani, M., van der Torre, L., Yorke-Smith, N.: Agent-Oriented Cooperative Smart Objects: From IoT System Design to Implementation. IEEE Transactions on Systems, Man, and Cybernetics: Systems **48**(11) (2017)
9. Ding, Z., Sun, Y., Jiang, C., Zhou, M., Liu, J., Song, W.: Performance Evaluation of Transactional Composite Web Services. IEEE Trans. Systems, Man, and Cybernetics: Systems **46**(8) (2016)
10. Fauvet, M., Duarte, H., Dumas, M., Benatallah, B.: Handling Transactional Properties in Web Service Composition. In: WISE'2005. New-York, USA (2005)
11. Frank, L., Ulslev Pedersen, R.: Integrated Distributed/Mobile Logistics Management. Transactions on Large-Scale Data- and Knowledge-Centered Systems **5** (2012)
12. Issarny, V., Bouloukakis, G., Georgantas, N., Billet, B.: Revisiting Service-oriented Architecture for the IoT: a Middleware Perspective. In: ICSOC'2016. Banff, Alberta, Canada (2016)
13. Leppänen, T., Riekki, J.: A Lightweight Agent-based Architecture for the Internet of Things. In: WEICE'2013. Wuxi, China (2013)
14. Li, L., Chengfei, L., Junhu, W.: Deriving Transactional Properties of Composite Web Services. In: ICWS'2007. Salt Lake City, Utah, USA (2007)
15. Little, M.: Transactions and Web Services. Communications of the ACM **46**(10) (2003)
16. Ma, J., Cao, J., Zhang, Y.: Efficiently Supporting Secure and Reliable Collaboration in Scientific Workflows. Journal of Computer and System Sciences **76**(6) (2010)
17. Maamar, Z., Baker, T., Sellami, M., Asim, M., Ugljanin, E., Faci, N.: Cloud *versus* Edge: Who Serves the Internet-of-Things Better? Internet Technology Letters, Wiley **1**(5) (2018)
18. Maamar, Z., Narendra, N., Benslimane, D., Sattanathan, S.: Policies for Context-driven Transactional Web Services. In: CAiSE'2007. Trondheim, Norway (2007)
19. Mihailescu, R., Davidsson, P., Spalazzese, R., Heyer, C.: A Role-Based Approach for Orchestrating Emergent Configurations in the Internet of Things. In: IoA'2017. São Paulo, Brazil (2017)
20. Mzahm, A., Ahmad, M., Tang, A.: Agents of Things (AoT): An intelligent operational concept of the Internet of Things (IoT). In: ISDA'2013. Bangi, Malaysia (2013)
21. Vidyasankar, K.: Transactional Properties of Compositions of Internet of Things Services. In: ISC2'2015. Guadalajara, Mexico (2015)
22. Vidyasankar, K.: A Transaction Model for Executions of Compositions of Internet of Things Services. In: Affiliated Workshops with ANT'2016 and SEIT'2016. Madrid, Spain (2016)
23. Zhang, M., Nodine, M., Bukhres, O.: Ensuring Relaxed Atomicity for Flexible Transactions in Multidatabase Systems. Tech. rep., Purdue University, CSD-TR-93-069 (November 1993)