Towards Enabling Internet-Scale Context-as-a-Service: A Position Paper

Alexandru Sorici
University Politehnica of Bucharest
Bucharest, Romania
alexandru.sorici@cs.pub.ro

Andrei Olaru
University Politehnica of Bucharest
Bucharest, Romania
andrei.olaru@cs.pub.ro

Adina Magda Florea
University Politehnica of Bucharest
Bucharest, Romania
adina.florea@cs.pub.ro

ABSTRACT

Deploying context management systems at a global scale comes with a number of challenges and requirements. We argue that the hypermedia model and the agent-oriented paradigm help achieve the vision of Context-as-a-Service. We categorize challenges according to context processing concerns and use a scenario to exemplify how the proposed architectural principles help overcome the challenges.

KEYWORDS

context management, context-awareness, context-as-a-service, hypermedia, software agents

ACM Reference Format:
Alexandru Sorici, Andrei Olaru, and Adina Magda Florea. 2019. Towards Enabling Internet-Scale Context-as-a-Service: A Position Paper. In *Companion Proceedings of the 2019 World Wide Web Conference (WWW '19 Companion), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3308560.3316511

1 INTRODUCTION

The observation of an exponential increase of internet-connected sensors and actuators has been a continuous remark for the past decade in the Internet-of-Things (IoT) community, as well as at the level of the everyday consumer of internet-enabled services. While true in terms of the increase of devices, the global-scale exploitation of the wealth of data brought forth by such devices still trends much behind.

Research and industry communities alike have invested in three main areas of focus: industry, society and environment. Of these, the societal focus area is mostly consumer-oriented and, as such, is the one by which the general public can perceive the advancement of the field. While there has been steady progress, the vision held by the ISTAG group back in 2001, in its proposed scenarios [5], where IoT and Ambient Intelligence (AmI) meet, is still far from reach.

In the Maria scenario [5], for example, a single personal assistant, running on a smartphone, handles interactions with transportation, security, communication and smart home systems. The complexity of these interactions comes from the need to continuously and seamlessly switch between different *contexts*.

WWW '19 Companion, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6675-5/19/05.
https://doi.org/10.1145/3308560.3316511

Context "is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves" [1]. Another definition [2] states that context is the *dressing* of a focus, separating the focus of the user or the application from the "dressing" – data which is not vital, but which can improve reasoning on the focus. While these definitions look general, they also offer an explanation for the complexity of deploying the Maria scenario.

Context-aware applications require reasoning procedures, context information representation and support for dynamic and heterogeneous context providers and consumers, all of which can vary greatly across domains. For instance, in a smart crop monitoring application, context refers to the data retrieved from *static* sensors installed in the field, for example monitoring soil and weather conditions, as well as meta-data about the sensors themselves (battery power, physical location, distance from other sensors). On the other hand, in a smart city application, user focus may change rapidly from the current activity at home or at the workplace, to that of monitoring personal health parameters while executing a physical workout activity. The consumer of context is mobile and changes focus quickly, while the means to derive the required high-level context information may involve data-driven algorithms (machine-learned data), along-side knowledge-driven heuristics.

Apart from the variety of context-aware application domains themselves, the problem of interest for *specific* context information within a domain, or combining information from several domains gave rise to the Sensing-as-a-Service model for IoT [13].

The model tackles the issue of siloed systems, proposing the existence of several stakeholders: the *sensor owner* (e.g. an individual person, a state or a private organization), the *sensor publishers* (organizations implementing means for sensor data collection), the *extended service providers* (organizations that bring added value by analyzing and aggregating sensor data) and the *final consumer*.

The sensing-as-a-service vision carries with it implicit non-functional requirements for *openness* and *scalability*, since the model allows for and encourages organizations to play several of the mentioned roles at once, thus leading to the development of *context prosumers* (producer *and* consumer at the same time). Consequently, context management plays a central role in this vision, precisely because it implies that consumers, with *different interests*, require data from different providers (with their own interests), processed in *custom* ways.

From a context management perspective, implementing the sensing-as-a-service model, as well as the ISTAG vision for AmI, brings about a number of challenges, which we group along the following concerns.

Representation and Reasoning. At global scale, a single context model or reasoning/inference technology cannot be expected for all application domains.Mechanisms that facilitate conversion from one model to another, as well as on-demand execution of various context processing procedures have to be considered.

Context Provisioning. This perspective focuses on how context information is supplied to services that can add value to raw data through inference. The challenge in AmI and sensing-as-a-service lies in the fact that the input base can no longer be considered static. Rather, a *long-lived* means to *interconnect*, *search for* and *select* the most appropriate/available context sources and context processors required for a consumer request is needed.

Query and Dissemination. The large-scale nature of sensing-as-a-service and pervasive AmI means that there must be a means to handle queries/subscriptions with a large selection base. Handling of frequently overlapping queries, as well as result caching should be enabled where appropriate. Another challenge comes from the principle that context information should be consumed as *close* as possible to its production source, thereby increasing the chance of relevance. An organizational scheme is needed that facilitates *local* consumption of context information, while at the same time enabling a *structured dissemination* of context information for remote consumers.

Context Service Deployment. The challenge from the deployment perspective is to find the means to facilitate scalability, search and discoverability of context management services. An organizational scheme that facilitates a *single entry-point* for all context life cycle entities (producers, processors, consumers) is required. The deployment structure must address mobility of context consumers and, more importantly, the *shift* in their focus, which brings a change in the necessary context information.

While these challenges have been addressed individually in existing work, we argue that visions such as sensing-as-a-service and ISTAG-level AmI are still not realized because of two main short-comings: **(i)** the lack of support architecture and information structuring mechanisms in existing context middleware that would allow for organization of context such that it becomes discoverable / searchable at a large scale, by highly mobile consumers; **(ii)** insufficient support in existing context middleware to *negotiate* between different (possibly conflicting) stakeholder (e.g. context producer, processor, consumer) interests, when considering the sensing-as-a-service model.

To answer to these challenges, we argue that context management middleware must adopt an architectural style that promotes long-lived deployment, openness and evolvability. Specifically, we posit that the fundamental engineering techniques that sustain the Web (e.g. hypermedia-driven interactions, RESTful [6] protocols, knowledge graphs, publish/subscribe mechanisms) can be coupled with the organizational and behavioral principles stemming from the Multi-Agent System (MAS) literature to obtain architectural specifications for Context-as-a-Service (CaaS) systems supporting application areas as large-scale as AmI and Sensing-as-a-Service. While a hypermedia environment facilitates large scale discoverability/searchability, the agent-oriented paradigm allows conceptualization of the sensing-as-a-service stakeholders as autonomous entities upholding policy driven goals.

2 RELATED WORK

Recent survey papers [10, 12] perform a good review of several of the existing context management middleware. While each solution covers many aspects that are essential for the implementation of a context management life cycle, most of the existing proposals either focus on specializing for a particular application domain (e.g. SeCoMan [3] – location, CoCaMAAL [7] – eHealth), or on the engineering effort for one or more of the focus requirements outlined in the introduction. The CoaaS platform introduced in [8], for example, introduces a detailed view of architectural elements responding to challenges of on-demand reasoning and large scale query management (including caching and request prediction). However, the systems proposes a centralized, cloud-based deployment and does not address the issue of organizing context information and searchability/discoverability thereof, beyond a directory-based mechanism.

CONSERT [14] is a context middleware that uses the multi-agent programming paradigm to design autonomous management for each of the context life cycle entities. It further proposes a deployment mechanism that follows an explicit organizational scheme, which uses *Context Dimensions* (well-defined focus points of a consumer that dominate the rest of the perceived context information - e.g. location, activity, role in an organization) to connect the different instances of context agents. While conceptually it answers to the challenges enumerated under the query/dissemination and service deployment categories, it does not meet the requirements for context provisioning and multi-modal, on-demand reasoning capabilities. Furthermore, the middleware has not undergone any real-world scenario validation.

FIWARE[1] is an open-source cloud platform aiming to provide reliable means for developing open, collaborative and mature ecosystems of smart, context-aware, internet-scale applications. It comes close to the set of requirements outlined in the introduction.

The context management reference architecture of FIWARE[2] is based on the interaction between key Generic Enablers (GE) that facilitate the development of a context processing pipeline. The central element is the *Context Broker* GE, which defines the context model and allows for context information storage, update and look-up. The *Context Broker* can be connected to a series of other GEs which complement its functionality (e.g. a *IoT Broker* as a wrapper over IoT devices, *IoT Discovery* to facilitate discovery of device capabilities, *Complex Event Processing* and *Big Data Analysis* GEs for short- and long-term reasoning). A core feature of the FIWARE platform is its reliance on web-based standards for enabling the communication between GEs. The Open Mobile Alliance (OMA) NGSI-9 and NGSI-10 [9] are two specifications for RESTful interaction protocols that define means to semantically describe the capabilities of IoT devices (NGSI-9) and the exchange of context content itself (e.g. publishing, updating, querying – through NGSI-10).

However, while the structural elements are there, the FIWARE platform proposes no conceptual means of organizing the connection of its GEs, so as to enable large-scale, automated discoverability

[1]FIWARE (https://www.fiware.org/, https://www.fiware.org/developers/catalogue/)
[2]FIWARE Context Management Architecture (https://goo.gl/Yvwv5A)

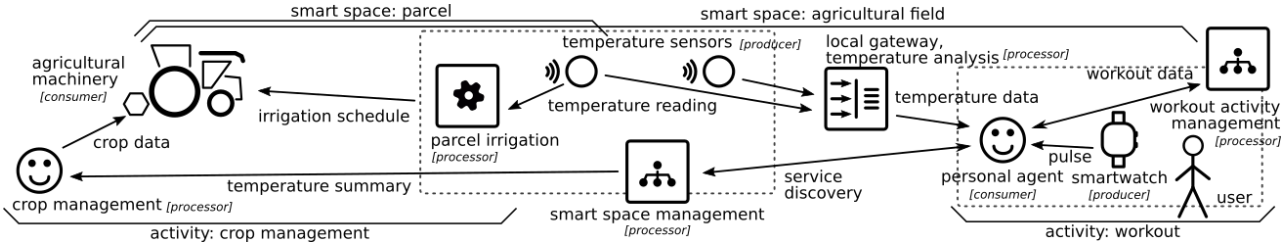

Figure 1: An example of context management services enabling a scenario in which temperature data is used both for crop management and for a user's context-aware activity tracking. Context life cycle entity roles are written in parentheses for each actor. Dotted boundaries denote logical partition of context life cycle entity deployment into *Context Domains* (e.g. parcel, workout), based on spatial and activity *Context Dimensions*.

and dissemination of information. The *Context Broker* interconnection has to be manually specified and is predefined at development time. Furthermore, the existing *Context Broker* implementation does not handle complex (composite) query execution, nor can the *Complext Event Processing* GE manage on-demand context reasoning.

Nonetheless, the takeaway point of FIWARE is that the defined RESTful interfaces contain in their specification the potential to enable all the outlined types of interaction, as required. Either upgrading the existing GEs, or delivering fresh implementations compatible with the NGSI-9 and NGSI-10 specifications has the potential to overcome the challenges.

3 HYPERMEDIA-DRIVEN CAAS

While the exact mechanisms and methods for context management are highly application-dependent, the multitude of entities and stakeholders in a global-scale system for the management of context information requires a suitable underlying architecture, which can answer to the outlined challenges and also fulfill the non-functional requirements of openness and scalability. The purpose of this architectural specification is to lead to the development of systems that support a large-scale *Context-as-a-Service* (CaaS) view, where context management can be offloaded to a network of *context life cycle entities*, each working under its own policy on context production, reasoning, or query management/dissemination.

Such an underlying architecture is the *resource-oriented* model of the Web and the *hypermedia-driven interactions* it supports. Moreover, the *agent-oriented paradigm* can contribute to modeling individual entities in the CaaS ecosystem, by focusing on a perspective centered on individual participants, rather than on the system as a whole.

Hypermedia underpins the World Wide Web as a network of uniquely identifiable (by means of URIs) *resources* interconnected through *web services*. However, to truly exploit the resource-oriented nature for the modeling of context processing entities, their *state* (e.g. context production capabilities, query subscription results, active reasoning mechanisms) has to be explicitly (semantically) described and the means to *change their state* should be clearly identifiable.

In terms of the **context service deployment** challenges, the use of a hypermedia environment and RESTful [6] interactions is not sufficient on its own to facilitate discoverability and efficient search / query propagation. In the architectural specification we envision, context life cycle entities must have a structured means by which to determine their *required* connections. We draw inspiration from the vision of Socio-Technical Networks (STNs) [4] and the proposed deployment organization scheme of CONSERT [14]. All URIs referring to connections between entities of an STN are *typed relations*, meaning that there is a clear semantics attached to them (e.g. ownership, membership, collocation). In CONSERT, *Context Dimensions* and *Context Domains* are concepts defining key *properties* and *values* that relate a context consumer to context sources that might be of interest to his current *focus*. For instance, the focus of the user may be included in various *spatial Context Domains*, also a hierarchy of *activity*-related *Domains*, and also in some *social Domains*. This hierarchical organization of context entities in the same *Dimension* facilitates their management and indicates clearly the context sub-model of which they are responsible.

Let us take the example in Figure 1. A person is taking a jog (workout) in the vicinity of a farm. Two dimensions of context are distinguishable: a *spatial* one and an *activity*-based one. From the spatial perspective, the ⟨`smart space context manager`⟩ keeps track of a hierarchy of two domains – a particular *parcel*, which is ⟨`part-of`⟩ an *agricultural field*. In the context model, an activity has a place where it is carried out, a relation which is modelled semantically and explicitly by the ⟨`personal agent`⟩ (which acts as a *context consumer*) on behalf of the jogger. By this relation, the ⟨`personal agent`⟩ uses a discovery mechanism through which it looks explicitly for a *context processor* responsible for the space in which the jogging activity takes place. The ⟨`personal agent`⟩ then launches a query for the current surface temperature. The result of this query is forwarded to the ⟨`workout activity manager`⟩, where this *context processor* uses it, together with data from the smart watch, to warn the user of excess dehydration, depending on the outside weather conditions and the calories he has burned.

The example illustrates both the model of sensing-as-a-service, since various information is gathered from various sources in the current context; and the model of structured search for context information, since entities are communicating with other entities in the same *Context Domain*.

The critical observation regarding the *deployment perspective* in the given example is that the ⟨`personal agent`⟩ *is* or *becomes* aware of two *Context Domains* (the jogging activity and the agricultural field location) and is able to connect to the relevant *context processors* to retrieve context that "dresses" the current focus of the user (his

well-being while jogging). This is made possible by the explicit, *typed* spatial inclusion and current activity relations which relate the ⟨personal agent⟩ with the ⟨workout activity manager⟩ / ⟨smart space manager⟩.

From a **query / dissemination perspective**, a *Context Dimension*-based organization of *context processors* facilitates query routing. In our example, a manner of handling queries similar to the one described by Mayer et al. [11] is proposed. Both the agricultural machinery and the personal agent subscribe for temperature updates to the ⟨temperature analysis⟩ *processor*. Being mobile consumers, they will request temperature updates for given locations (parcels) within the field. If temperature analysis is managed by different *processors*, queries sent to the processor for the entire field must be routed to the one corresponding to current consumer location. The *Context Dimension* / *Context Domain* based organization of the *processors* enables the same type of query routing mechanism as described in [11] (including *Context Domain*-based range queries).

An *agent-oriented view* of the context life cycle elements in our example is of relevance from the **context provisioning perspective**. Software agents are entities that are able to act autonomously in their environment in order to achieve or maintain their goals.

The owner of the agricultural field is the one that deployed the temperature sensors and the ⟨temperature analysis⟩ *processor(s)*. The ⟨personal agent⟩ may be able to discover the existence of the *processor* but this may not guarantee that he has the *right* to access their information. This use case can be viewed as a sensing-as-a-service instance, where the ⟨temperature analysis⟩ is an *extended service provider* and the ⟨personal agent⟩ a *final consumer*. If we model/design the life cycle entities as *agents*, then access to the temperature data can be subject to an automated, agent based negotiation. The ⟨temperature analysis⟩ *processor* can make use of goal-based policies, that require it to, for example, accept query requests only for a specific user activity context, such as the workout one.

The **representation and reasoning perspective** benefits in two ways from the proposed principles. Context models are application dependent, but the desire for searchability requires the use of a common-ground language. Standardization efforts of the W3C such as the SSN ontology[3] or the Web-of-Things Thing Description[4] can be employed in this case. The agent-oriented view, on the other hand, is suitable for multi-modal and on-demand reasoning. In our example, default reasoning carried out by the ⟨workout activity⟩ *processor* is complemented (upon discovery of temperature data) by a request from the ⟨personal agent⟩. The ⟨workout activity⟩ is asked to execute a knowledge-driven heuristic that combines temperature *and* smart watch context to gauge the user's well-being.

4 CONCLUSIONS

We argue that context servicing at the scale of pervasive AmI settings and sensing-as-a-service applications is lacking because of a missing evolvable and searchable middleware ecosystem, where organized context information enables its effective exploration. We further assess that an agent-based view of the context life cycle entities can appropriately model the required level of autonomy and goal-driven behavior of stakeholders in a sensing-as-a-service setting.

We envision a large-scale CaaS deployment as a conglomerate of computing entities obtaining producing, processing, and consuming context. The multitude and diversity of entities inevitably leads to a highly *heterogeneous* system, that needs to be *loosely coupled*. Not only does the hypermedia model naturally support openness and scalability, but it also answers well to the challenges of CaaS deployment. We also argue that context management systems would benefit from an agent-oriented approach to the implementation of their individual components. What we can take from the MAS domain and use in CaaS is not necessarily how entities can be implemented as agents, but the *agent-oriented perspective*. Moving from a system-wide view to an entity-centered helps designing components that are able to work in highly heterogeneous systems.

[3]https://www.w3.org/TR/vocab-ssn/

[4]https://w3c.github.io/wot-thing-description/

ACKNOWLEDGMENTS

This research was funded by grant PN-III-P1-1.2-PCCDI-2017-0734.

REFERENCES

[1] Gregory D Abowd, Anind K Dey, Peter J Brown, Nigel Davies, Mark Smith, and Pete Steggles. 1999. Towards a better understanding of context and context-awareness. In *International symposium on handheld and ubiquitous computing*. Springer, 304–307.

[2] Juliette Brézillon and Patrick Brézillon. 2007. Context Modeling: Context as a Dressing of a Focus. In *Modeling and Using Context*, Boicho Kokinov, Daniel C. Richardson, Thomas R. Roth-Berghofer, and Laure Vieu (Eds.). Lecture Notes in Computer Science, Vol. 4635. Springer Berlin Heidelberg, 136–149.

[3] Alberto Huertas Celdrán, Félix J García Clemente, Manuel Gil Pérez, and Gregorio Martínez Pérez. 2016. SeCoMan: A Semantic-Aware Policy Framework for Developing Privacy-Preserving and Context-Aware Smart Applications. *IEEE Systems Journal* 10, 3 (2016), 1111–1124.

[4] Andrei Ciortea, Antoine Zimmermann, Olivier Boissier, and Adina Magda Florea. 2016. Hypermedia-driven Socio-technical Networks for Goal-driven Discovery in the Web of Things. In *Proceedings of the Seventh International Workshop on the Web of Things*. ACM, 25–30.

[5] K. Ducatel, M. Bogdanowicz, F. Scapolo, J. Leijten, and J.C. Burgelman. 2001. *Scenarios for ambient intelligence in 2010.* Technical Report. Office for Official Publications of the European Communities.

[6] Roy T Fielding and Richard N Taylor. 2000. *Architectural styles and the design of network-based software architectures*. Vol. 7. University of California, Irvine Irvine, USA.

[7] Abdur Forkan, Ibrahim Khalil, and Zahir Tari. 2014. CoCaMAAL: A cloud-oriented context-aware middleware in ambient assisted living. *Future Generation Computer Systems* 35 (2014), 114–127.

[8] Alireza Hassani, Alexey Medvedev, Pari Delir Haghighi, Sea Ling, Maria Indrawan-Santiago, Arkady Zaslavsky, and Prem Prakash Jayaraman. 2018. Context-as-a-Service Platform: Exchange and Share Context in an IoT Ecosystem. In *2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE, 385–390.

[9] Srdjan Krco, Boris Pokric, and Francois Carrez. 2014. Designing IoT architecture(s): A European perspective. In *Internet of Things (WF-IoT), 2014 IEEE World Forum on*. IEEE, 79–84.

[10] Xin Li, Martina Eckert, José-Fernán Martinez, and Gregorio Rubio. 2015. Context aware middleware architectures: survey and challenges. *Sensors* 15, 8 (2015), 20570–20607.

[11] Simon Mayer, Dominique Guinard, and Vlad Trifa. 2012. Searching in a web-based infrastructure for smart things. In *Internet of Things (IOT), 2012 3rd International Conference on the*. IEEE, 119–126.

[12] Charith Perera, Arkady Zaslavsky, Peter Christen, and Dimitrios Georgakopoulos. 2014. Context aware computing for the internet of things: A survey. *IEEE communications surveys & tutorials* 16, 1 (2014), 414–454.

[13] Charith Perera, Arkady Zaslavsky, Peter Christen, and Dimitrios Georgakopoulos. 2014. Sensing as a service model for smart cities supported by internet of things. *Transactions on Emerging Telecommunications Technologies* 25, 1 (2014), 81–93.

[14] Alexandru Sorici, Gauthier Picard, Olivier Boissier, and Adina Florea. 2015. Multi-agent based flexible deployment of context management in ambient intelligence applications. In *International Conference on Practical Applications of Agents and Multi-Agent Systems*. Springer, 225–239.