A Generic Framework for Quality-Based Autonomic Adaptation Within Sensor-Based Systems

Antoine Auger[1(✉)], Ernesto Exposito[2], and Emmanuel Lochin[1]

[1] Institut Supérieur de l'Aéronautique et de l'Espace (ISAE-SUPAERO), Université de Toulouse, 31055 Toulouse Cedex 4, France
{antoine.auger,emmanuel.lochin}@isae.fr
[2] Laboratoire Informatique de l'Université de Pau et des Pays de l'Adour (LIUPPA), Anglet, France
ernesto.exposito@univ-pau.fr

Abstract. With the growth of the Internet of Things (IoT), sensor-based systems deal with heterogeneous sources, which produce heterogeneous observations of disparate quality. Since network QoS is rarely sufficient to expertise Quality of Observation (QoO), managing such diversity at the application level is a very complex task and requires high levels of experience from application developers. Given this statement, this paper proposes a generic framework for QoO-based autonomic adaptation within sensor-based systems. An abstract architecture is first introduced, intended to bridge the gap between sensors capabilities and application needs thanks to the Autonomic Computing paradigm. Then, the framework is instantiated and practical considerations when implementing an autonomous sensor-based system are given. We illustrate this instantiation with concrete examples of sensor middlewares and IoT platforms.

Keywords: Internet of things · Sensors · Observations · Information quality · Adaptation · Autonomic computing · Service-oriented architectures

1 Introduction

With the accelerated development of Internet of Things (IoT), more and more information sources are available to applications, generally through intermediate sensor-based systems such as sensor middlewares [9,15] or IoT platforms [25]. Whether physical or virtual, sensors represent a huge opportunity for collecting and processing information to provide enhanced services and improve the quality of life within cities.

Only few sensor-based systems have been designed to adapt their behavior according to application needs. Instead, these systems usually define their own metrics before delegating the management of observation quality to end

K. Drira et al. (Eds.): ICSOC 2016 Workshops, LNCS 10380, pp. 21–32, 2017.
https://doi.org/10.1007/978-3-319-68136-8_2

applications. However, this adaptation strategy has several drawbacks since it assumes that application developers have the knowledge to understand those metrics and their meaning. Furthermore, some metrics could be missing or poorly implemented.

On one hand, sensor-based systems deal with various sensors, which constitute as many heterogeneous sources that produce observations of disparate quality. These systems receive and process observations to enrich them with additional information or make them more meaningful for end applications. On the other hand, applications may have specific observation needs, in particular concerning the Quality of Observation (QoO). These needs, which differ from one application to another, may be dynamic and therefore vary over time. For instance, two applications may ask for the same kind of observations (temperature, wind, etc.) but may not require the same granularity (e.g., frequency, coverage).

In order to bridge the gap between heterogeneous observations and application-specific QoO needs, we envision dynamic adaptation with only few interventions from developers/experts. By following the Autonomic Computing [12] approach, we propose sensor-based systems to play the role of autonomic mediators that adapt their behavior to fit application needs thanks to the definition of Service Level Agreements (SLAs). In this paper, a generic framework for QoO-based autonomic adaptation will be presented. This framework, suitable to a large number of sensor-based systems, consists in two parts: (i) an abstract architecture composed of different layers for observation consumption and (ii) a description of five autonomic maturity levels from a sensor perspective.

The remainder of this paper is organized as follows. Section 2 introduces the required background. Section 3 presents our contribution which consists in a generic framework. Section 4 focuses on framework instantiation, giving concrete implementation examples from sensor middlewares and IoT platforms. Finally, we present existing and relevant work in Sect. 5 before concluding and giving some research perspectives in Sect. 6.

2 Required Background

This Section aims to give the reader the required background about sensor observations, QoO and the Autonomic Computing paradigm. These notions are the three foundations of the framework that will be described later in Sect. 3.

2.1 Terminology for Sensor Observations

Sensor-based systems provide observations to applications. Each observation may be considered as the representation of a physical-observed phenomenon (the temperature of a place, a person that enters a room, etc.) or a virtual-occurred event (a new tweet from someone, an incoming e-mail, the availability of a new software update, etc.).

Previous studies have proposed taxonomies to denote the different types of observations that applications can consume. Indeed, the same phenomenon or

event can be reported in different ways, including more or less details about the unit of the measure, sensor type, location, etc. As a matter of fact, these taxonomies use ladder representations to denote the different observation types. For instance, in 2013, the National Institute of Standards and Technology (NIST) conducted a sensor ontology literature review [7] that introduced "raw data", "primitive" and "object" perception levels. More recently, Sheth has proposed the "data, information, knowledge, and wisdom (DIKW) ladder" for the IoT [23]. Such taxonomies aim to estimate the level of complexity required to process and "understand" them by observation consumers (applications or users).

In this paper, we only envision applications as observation consumers. Reusing existing terminologies, we define three levels for observation consumption that we define as follows, from the most basic to the most complex:

Sensor Raw Data. The first observation level corresponds to unprocessed observations coming from sensors. At this level, these observations are encoded in the key/value form and do not contain additional information. We denote them as sensor Raw Data (e.g., {`sensor_id: 34, value: 20`}).

Sensor Information. The second observation level corresponds to sensor Information. Sensor Information is sensor Raw Data that has been processed or enriched with additional Context information [17] (e.g., {`sensor_id: 34, value: 20, unit: Celsius, location: (43.564509, 1.468910), accuracy: 0.8`}).

Sensor Knowledge. The third observation level is reached with the use of semantics. By implementing a semantic annotation process, sensor-based systems are able to model domain-specific observations and thus to deal with machine-understandable information. We denote by sensor Knowledge any semantic-based observation representation (e.g., {`sensor_type: temperature, value: comfort, location: room3, accuracy: good`}).

These three observation types mainly refer to observation representation (broadly speaking their content and their format). In the next Section, we introduce attributes to characterize them more precisely and assess their intrinsic quality.

2.2 Quality of Observation

Since the goal of our framework is to provide application-specific high-quality observations, we present some popular quality dimensions and examples of metrics, which may later serve as a basis for SLAs' definition:

Data Quality (DQ). In this work, we use DQ notion to refer to the intrinsic quality of sensor Raw Data. Broadly speaking, it can be seen as the distance between the sensed value (the observation) and the corresponding event (the occurred event). Quite complex to assess, DQ is mainly impacted by the sensor device quality and performances of the underlying network.

Quality of Information (QoI). In [2], Bisdikian et al. define QoI as "*the collective effect of information characteristics (or attributes) that determine the*

degree by which the information is (or perceived to be) fit-to-use for a purpose". Some examples of QoI attributes are latency, reputation and spatio-temporal Context. In a sensor context, these metrics may be particularly useful for an application to assess more accurately how fit-for-use sensor Information is.

Quality of Context (QoC). According to Dey, Context can be defined as "*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*" [6]. QoC denotes Quality of Information applied to Context [3].

Quality of Knowledge (QoK). Some sensor-based systems, such as Semantic Sensor Webs [24], use ontologies to model sensor observations and describe the capabilities of their sensors. This semantic representation allows query inference and high-level reasoning on sensor Knowledge. QoK assesses the quality of the ontology-based modelisation. Some metrics (such as completeness, coverage and ease to use) have been proposed for Knowledge management systems [20] and may be applied to sensor Knowledge.

Quality of Service (QoS). QoS has been defined in the E.800 recommendation [10] by the ITU-T. Although this definition encompasses above quality dimensions, it is not the case in practice. Indeed, the term "Quality of Service" generally refers to packet transportation from source to destination within the network. As a matter of fact, the set of QoS metrics is often restricted to bandwidth, delay, jitter and loss probability. We will denote this quality dimension as "network QoS" in the rest of this paper.

Above quality dimensions may be used to characterize the general quality of sensor observations. Depending on sensors, applications and use cases, it may be relevant to use many of them to improve this characterization process. For instance, using both network QoS and QoI, one can better understand if some outdated observations are the result of poor network performances or due to a sensor sampling rate too low. In this paper, independently of their consumption level, we use the generic term "Quality of Observation" (QoO) to denote observation quality that sensor-based systems provide to upper applications.

2.3 Autonomic Computing Paradigm

Autonomic Computing has been defined by IBM as the ability of systems to "*manage themselves given high-level objectives from administrators*" [12]. Since it makes a clear distinction between goals and means, the Autonomic Computing paradigm is commonly considered as a convenient way to build interoperable, evolving and easy-to-use systems.

Autonomic systems are a set of Autonomic Elements. Each Autonomic Element is composed of one or many Managed Elements controlled by a single Autonomic Manager. This entity continuously monitors the internal state of its different Managed Elements; then analyses this information; and finally takes appropriate decisions based on both its knowledge base and high-level objectives.

At last, these decisions are converted into actions and transmitted to appropriate Managed Elements for execution. These different steps form the MAPE-K adaptation control loop (Monitor, Analyse, Plan, Execute, Knowledge base), also denoted as "MAPE-K loop" in the rest of this paper.

Autonomic systems relieve end-users to manually implement logic to comply with their needs. Instead, users express their goals, leaving to one or many Autonomic Managers the task of managing the different Autonomic Elements. This process implements the required self-properties. In [12], IBM has identified four *self-** fundamental adaptation properties for autonomic systems, namely self-configuration, self-optimization, self-healing and self-protection.

3 Generic Framework Proposal

In this Section, we present our generic framework for quality-based adaptation. This framework describes an abstract architecture and describes five autonomic maturity levels for sensor-based systems.

3.1 Abstract Architecture

Figure 1 depicts a high-level representation of a sensor-based system providing QoO-based adaptation. Inspired by the representation of standard IT protocol stacks, it shows the different flows that exist between components: while solid arrows represent observation consumption flows, dashed arrows indicate adaptation-related flows (control, management, etc.).

As previously stated, we only consider three different types of observations that applications can consume. We denote as "Raw Data layer", "Information layer" and "Semantic layer" the layers at which applications can consume Raw Data, Information or Knowledge observations, respectively. In fact, one may see each layer as an abstract service provider offering collection and digitization, characterization and semantic annotation to upper layers, respectively. In the following, we give more details about these three layers:

Raw Data layer. The Raw Data layer deals with observations that are produced by sensors. Raw observations can either refer to phenomena (for physical sensors) or events (for virtual sensors). This layer offers collection and digitization of these observations to upper layers. Regarding physical sensors, the digitization process consists in the translation of observations from the physical world into the digital world (e.g., using sensor adapters/wrappers).

Information layer. The Information layer offers Raw Data characterization. The fact of annotating sensor Raw Data with Context information adds value to it since end applications can then use this Context information to assess QoO. For instance, sensor provenance, spatio-temporal information and sensor confidence level are some concrete Context attributes that can be added to sensor Raw Data.

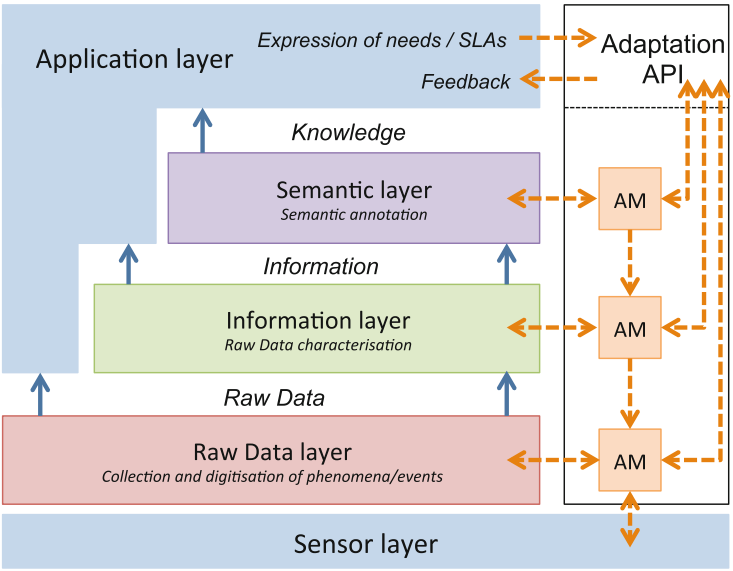

Fig. 1. Abstract architecture for application-specific QoO adaptation. AM: **A**utonomic **M**anager, SLAs: **S**ervice **L**evel **A**greements.

Semantic layer. The Semantic layer offers semantic-based annotation of sensor Information. The use of semantic-based representations (in general using ontologies) allows applications to consume machine-understandable Knowledge. Since ontologies are conceptual representations of a specific domain, we denote as Knowledge any semantic-based observation representation. Remember that this kind of observations allows high-level inference and reasoning.

According to the proposed architecture, applications can customize the behavior of the sensor-based system by expressing needs through the adaptation API. For instance, a classic application for environmental monitoring may be less sensitive to delay than another one for military battlefield management. Therefore, these two applications can ask for the same kind of observations (temperature, wind, etc.) by specifying different QoO levels (with the definition of SLAs). Then, these SLAs are routed to the appropriate Autonomic Manager(s). Each Autonomic Manager (AM) should take into account these application needs and add them to its Knowledge base. When the QoO needs are not or no longer fulfilled, the Autonomic Manager will enforce the required corrective actions to be implemented by the corresponding layer. In addition to possibly consuming observations at three different levels, applications can also subscribe to feedback (statuses of the different layers, statistics of the MAPE-K loops, QoO metrics, etc.) sent by the autonomic sensor-based system.

This framework considers each observation layer as a Managed Element, managed by its own Autonomic Manager for scalability reasons. To adapt QoO

according to application needs, we assume that each observation layer implements the required adaptation mechanisms. Some examples of these mechanisms are observation Filtering, Formatting, Fusion, Caching or Machine Learning. Please note that the study of these mechanisms is out of the scope of the framework.

3.2 Autonomic Maturity Levels for Sensor-Based Systems

With the presence of Autonomic Managers and adaptation-related flows, our abstract architecture falls within the Autonomic Computing paradigm. Inspired by the work of IBM on Autonomic Computing [11], we adapt and describe five maturity levels to quantify *how autonomous an existing sensor-based system is*:

Basic (level 1). At this level, no customization is available for applications. The entire behavior of the system is hard-coded by developers during the design phase. The monitoring of the system is done manually by developers who also replace or update different elements and components accordingly.

Managed (level 2). At this autonomic level, adaptation is based on predefined rules written by developers or domain experts (e.g., meteorologists). These rules are simple (*if value < 0 then drop(wind_speed_observation)* for instance) and are generally written by a skilled person.

Predictive (level 3). Predictive behavior is reached with the implementation of reasoning processes in some components of the sensor-based system. These processes can consist in Fusion, Machine Learning, etc. but they must not take into account any macroscopic goal. At this maturity level, components are generally selfish entities.

Adaptive (level 4). Adaptive behavior is characterized by the definition of SLAs. SLAs mostly correspond to the definition of application profiles with specific QoO needs. At this maturity level, components take into account SLAs to self-adapt their local behavior. By implementing such mechanisms, a sensor-based system can be seen as a system driven by a macroscopic goal.

Autonomic (level 5). The last maturity level is the autonomic one. A sensor-based system may be considered as autonomic when its behavior is driven by the expression of business rules coming from end applications. Such system automatically derives appropriate SLAs from these rules and routes them to its different Autonomic Elements. Then, these autonomic entities accordingly adapt their behavior and collectively fulfill application needs.

To fully take advantage of the Autonomic Computing paradigm, we recommend to instantiate our framework to build adaptive or autonomic sensor-based systems (level 4 or 5) with definition of SLAs. These SLAs should include both observation needs (e.g., {`type: temperature, level: information`}) and the QoO level required by the application (e.g., {`timeliness: 60, trust: 0.8`}).

Since it is composed of generic components (conceptual layers), this framework can be adapted to a large number of platform-specific implementations. For example, one could imagine a sensor-based system where sensors semantically

annotate their observations. In this case, the Sensor, Raw Data, Information and Semantic layers would be only one. Regarding adaptation, our framework is also generic about the chosen autonomic maturity level, the SLAs definition and the available adaptation mechanisms.

4 Framework Instantiation

In this Section, we focus on practical considerations when implementing an autonomous sensor-based system. QoO metrics, mediation and SLAs are required by our framework in order to provide application-specific QoO adaptation. We illustrate each of these features by giving concrete examples of existing sensor-based systems. These examples aim to instantiate our framework and validate the abstract architecture previously introduced.

4.1 QoO Metrics Definition

In Sect. 2, we have seen that numerous quality dimensions may be considered to derive metrics and define new SLAs.

Some sensor-based systems define their own custom QoO metrics. For instance, the MASTAQ middleware [9] proposes "standard deviation" and "confidence level" as QoO metrics while MiddleWhere [19] defines "resolution", "confidence" and "freshness" metrics to assess the quality of location information. Other sensor-based systems use frameworks to define QoO metrics. For instance, INCOME [15] is a QoC-based middleware for Context distribution. It allows Context consumers to express SLAs according to their QoC needs. The distribution of Context information is then performed according to these needs. In order to model heterogeneous QoC metrics, INCOME uses the QoCIM framework [14].

Although it allows finer metric tuning, defining custom metrics may decrease the system interoperability. Indeed, a simple name is rarely sufficient to understand how a metric is computed or how it should be used. As a result, sensor-based systems that define their own metrics must provide adequate documentation, which details and clarifies the metrics used to avoid any ambiguities.

4.2 Mediation

Mediation feature has gained attention with Service-Oriented Architectures. A Service-Oriented Architecture (SOA) is an architectural framework for building software systems based on distributed services which may be offered by different service providers. SOA software architectures are based, among other things, on the key concepts of service, service provider and service consumer [13]. A service is a well-defined and self-contained function or functionality offered by a service producer. Service consumers are the entities that make use of the services provided. Sometimes, the use of a service must respect a SLA, which specifies the purpose, functionalities, constraints and usage of the service.

Sensor-based systems often play the role of mediators, bridging the gap between sensors and applications. Even if all sensor-based systems are not built following SOA framework, the underlying sensors may be considered as "observation providers" while upper applications can be seen as "observation consumers". In some cases, it may occur that several sensors have similar capabilities. Although they offer the same kind of observation, they may not offer the same QoO level. To optimise resources, some sensor-based systems only select a subset of service providers that are sufficient to fill customer needs. This mechanism is called sensor composition/selection and it has gained popularity with Semantic Web Services. For instance, in [18], Perera et al. propose CASSARAM, a Context-aware tool to select an optimal subset of sensors according to specific QoO attributes (availability, accuracy, etc.).

Implementation choices have an effect on mediation feature. While a SOA-based system can rely on a service bus playing the role of mediator, other architectures (such as microservice-based architectures) should implement their own mediation mechanism.

4.3 Application Needs and Sensor Capabilities

In order to provide adaptation, our abstract architecture relies on the expression of application needs. This requirement supposes that (i) applications must be able to express their needs regarding observations and (ii) sensors must be able to express their capabilities and describe the characteristics of the service that they provide.

To cope with these challenges, several sensor-based systems have used semantics, and in particular ontologies, to model sensor observations and describe the capabilities of their sensors. Ontology-based representation involves the definition of concepts and their relationships. Using semantics for observation modeling corresponds to transform sensor Raw Data or sensor Information into machine-understandable Knowledge. Within semantic-based systems, sensors can also express their capabilities in a semantic way (what is their type, their sampling rate, their units, etc.).

Numerous sensors and observations ontologies have been developed, creating a need for standardisation. Between 2009 and 2011, the Semantic Sensor Network Incubator Group[1] of the W3C initiated a standardisation process. After reviewing 17 sensors and observations ontologies, they identified the most relevant concepts and developed the Semantic Sensor Network (SSN) ontology [5]. SSN ontology has been reused within CASSARAM middleware [18] for instance. In CASSARAM, sensors are semantically described using an extended SSN ontology while observations are semantically annotated with Context. Finally, OpenIoT project [25] also extends SSN ontology to generalize the notion of sensor, supporting both physical devices and virtual sensors.

[1] http://www.w3.org/2005/Incubator/ssn/.

Ontologies are efficient to build extensible, reusable and interoperable systems. The W3C SSN ontology is currently one of the most popular standard within sensor-based systems to both represent observations and describe sensor capabilities.

5 Related Work

QoS within Wireless Sensor Networks has been largely investigated [4]. Studies generally present network QoS as a way to improve general information quality within WSNs. By contrast, this assumption does not longer holds for other sensor-based systems such as IoT platforms. Indeed, within these observation-centric systems, network QoS is rarely sufficient to ensure observation quality. Therefore, other quality dimensions (like QoI) need to be considered. Until now, QoO has often been addressed thanks to Context-awareness feature [1,3]. Therefore, Context information and QoC have received much attention, especially in the areas of sensor middlewares [22] and IoT [17]. However, Context information is rarely used by sensor-based systems to perform autonomic adaptation but is rather added to sensor observations as meta-data for later analysis by applications.

Few research efforts have been made to manage QoO with Autonomic Computing. For instance, AcoMS [8] is a sensor middleware for Context distribution. It enables autonomic adaptation by providing several *self-** features such as configuration, reconfiguration and healing. The solution of Pathan et al. [16] allows sensor plug-and-play, as well as self-reconfiguration processes according to application scenarios and Context. Finally, SPACES [21] focuses on web-service adaptation by implementing an autonomic MAPE-K loop to dynamically change web services behavior according to Context information. Overall, the above solutions have highlighted the benefits to provide autonomic adaptation. However, they do not consider intrinsic QoO but only rely on the execution Context to define SLAs and provide adaptation. Finally, they do not take into account application-specific needs.

To the best of our knowledge, we are the first to propose a generic framework to achieve application-specific QoO adaptation within sensor-based systems.

6 Conclusion and Perspectives

In this paper, we introduce and describe a generic framework for autonomic adaptation within sensor-based systems relying on Quality of Observation (QoO). This framework aims to bridge the gap between sensors and applications thanks to the Autonomic Computing paradigm. In this paper, a layered architecture able to provide various levels of observation consumption according to application-specific QoO needs has been introduced. This abstract architecture relies on the definition of Service Level Agreements (SLAs) and can be instantiated according to different autonomic maturity levels.

By considering QoO, sensor-based systems go beyond commonly-used network QoS, which has shown its limitations within information-centric systems such as sensor middlewares and IoT platforms. We hope that our generic framework will help researchers and developers to build autonomous sensor-based systems that focus on their primary function, i.e., deliver high-quality observations to applications. Further experimentations are needed to estimate the relevancy of certain QOO metrics depending on use cases. Moreover, further investigation into MAPE-K autonomic control loop is strongly recommended, in particular concerning the Knowledge base (decision rules and execution plans).

As future work, we are currently developing a Cloud-based integration platform for QoI Assessment as a Service. Designed according to the framework presented in this paper, this platform will be applied to a Smart City use case. Such contribution will help the different Smart City stakeholders to assess, better understand and improve QoI in a collaborative way.

Acknowledgements. This research was supported in part by the French Ministry of Defense through financial support of the Direction Générale de l'Armement (DGA).

References

1. Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., Riboni, D.: A survey of context modelling and reasoning techniques. Pervasive Mob. Comput. **6**(2), 161–180 (2010)
2. Bisdikian, C., Branch, J., Leung, K., Young, R.: A letter soup for the quality of information in sensor networks. In: IEEE International Conference on Pervasive Computing and Communications, PerCom 2009, pp. 1–6, March 2009
3. Buchholz, T., Küpper, A., Schiffers, M.: Quality of context information: What it is and why we need it. In: Proceedings of the 10th HP-OVUA Workshop, vol. 2003 (2003)
4. Chen, D., Varshney, P.K.: QoS support in wireless sensor networks: a survey. In: International Conference on Wireless Networks, vol. 13244, pp. 227–233 (2004)
5. Compton, M., Barnaghi, P., Bermudez, L., García-Castro, R., Corcho, O., Cox, S., Graybeal, J., Hauswirth, M., Henson, C., Herzog, A.: The SSN ontology of the W3C semantic sensor network incubator group. Web Semant. Sci. Serv. Agents World Wide Web **17**, 25–32 (2012)
6. Dey, A.K.: Understanding and using context. Pers. Ubiquit. Comput. **5**(1), 4–7 (2001)
7. Eastman, R., Schlenoff, C., Balakirsky, S., Hong, T.: A Sensor Ontology Literature Review. Technical report NIST IR 7908, National Institute of Standards and Technology. http://nvlpubs.nist.gov/nistpubs/ir/2013/NIST.IR.7908.pdf
8. Hu, P., Indulska, J., Robinson, R.: An autonomic context management system for pervasive computing. In: Sixth Annual IEEE International Conference on Pervasive Computing and Communications, PerCom 2008, pp. 213–223, March 2008
9. Hwang, I., Han, Q., Misra, A.: MASTAQ: a middleware architecture for sensor applications with statistical quality constraints. In: Third IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom 2005 Workshops, pp. 390–395. IEEE (2005)

10. ITU-T: E.800: Definitions of terms related to quality of service. International Telecommunication Union-Telecommunication Standardisation Sector (ITU-T), September 2008
11. Jacob, B., Lanyon-Hogg, R., Nadgir, D.K., Yassin, A.F.: A practical guide to the IBM autonomic computing toolkit. IBM, International Technical Support Organization (2004)
12. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. Computer **36**(1), 41–50 (2003)
13. Krafzig, D., Banke, K., Slama, D.: Enterprise SOA: service-oriented architecture best practices. Prentice Hall Professional (2005)
14. Marie, P., Desprats, T., Chabridon, S., Sibilla, M.: The QoCIM framework: concepts and tools for quality of context management. In: Brézillon, P., Gonzalez, A.J. (eds.) Context in Computing, pp. 155–172. Springer, New York (2014). doi:10.1007/978-1-4939-1887-4_11
15. Marie, P., Lim, L., Manzoor, A., Chabridon, S., Conan, D., Desprats, T.: QoC-aware context data distribution in the internet of things. In: Proceedings of the 1st ACM Workshop on Middleware for Context-Aware Applications in the IoT, M4IOT 2014, pp. 13–18. ACM, New York (2014)
16. Pathan, M., Taylor, K., Compton, M.: Semantics-based plug-and-play configuration of sensor network services. In: SSN, Citeseer (2010)
17. Perera, C., Zaslavsky, A., Christen, P., Georgakopoulos, D.: Context aware computing for the internet of things: a survey. IEEE Commun. Surv. Tutorials **16**(1), 414–454 (2014)
18. Perera, C., Zaslavsky, A., Liu, C., Compton, M., Christen, P., Georgakopoulos, D.: Sensor search techniques for sensing as a service architecture for the internet of things. IEEE Sens. J. **14**(2), 406–420 (2014)
19. Ranganathan, A., Al-Muhtadi, J., Chetan, S., Campbell, R., Mickunas, M.D.: MiddleWhere: a middleware for location awareness in ubiquitous computing applications. In: Jacobsen, H.-A. (ed.) Middleware 2004. LNCS, vol. 3231, pp. 397–416. Springer, Heidelberg (2004). doi:10.1007/978-3-540-30229-2_21
20. Rao, L., Osei-Bryson, K.M.: Towards defining dimensions of knowledge systems quality. Expert Syst. Appl. **33**(2), 368–378 (2007)
21. Romero, D., Rouvoy, R., Seinturier, L., Chabridon, S., Conan, D., Pessemier, N.: Enabling context-aware web services: a middleware approach for ubiquitous environments. Enabling context-aware web services: methods, architectures, and technologies, pp. 113–135 (2010)
22. Sheikh, K., Wegdam, M., van Sinderen, M.: Middleware support for quality of context in pervasive context-aware systems. In: Fifth Annual IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops 2007, pp. 461–466, March 2007
23. Sheth, A.: Internet of things to smart IoT through semantic, cognitive, and perceptual computing. IEEE Intell. Syst. **31**(2), 108–112 (2016)
24. Sheth, A., Henson, C., Sahoo, S.: Semantic sensor web. IEEE Internet Comput. **12**(4), 78–83 (2008)
25. Soldatos, J., Kefalakis, N., Hauswirth, M., Serrano, M., Calbimonte, J.P., Riahi, M., Aberer, K., Jayaraman, P.P., Zaslavsky, A.: OpenIoT: Open Source Internet-of-Things in the Cloud (2015)