Survey on Quality of Observation within Sensor Web Systems*

Antoine Auger
Institut Supérieur de l'Aéronautique et de l'Espace (ISAE-SUPAERO),
Université de Toulouse, Toulouse, France
antoine.auger@isae.fr

Ernesto Exposito
Université de Pau et des Pays de l'Adour, LIUPPA, France

Emmanuel Lochin
Institut Supérieur de l'Aéronautique et de l'Espace (ISAE-SUPAERO),
Université de Toulouse, Toulouse, France

September 7, 2017

Abstract

The Sensor Web vision refers to the addition of a middleware layer between sensors and applications. To bridge the gap between these two layers, Sensor Web systems must deal with heterogeneous sources, which produce heterogeneous observations of disparate quality. Managing such diversity at the application level can be complex and requires high levels of expertise from application developers. Moreover, as an information-centric system, any Sensor Web should provide support for Quality of Observation (QoO) requirements. In practice, however, only few Sensor Webs provide satisfying QoO support and are able to deliver high-quality observations to end consumers in a specific manner.

This survey aims to study why and how observation quality should be addressed in Sensor Webs. It proposes three original contributions. First, it provides important insights into quality dimensions and proposes to use the QoO notion to deal with information quality within Sensor Webs. Second, it proposes a QoO-oriented review of 29 Sensor Web solutions developed between 2003 and 2016, as well as a custom taxonomy to characterise some of their features from a QoO perspective. Finally, it draws four major requirements required to build future adaptive and QoO-aware Sensor Web solutions.

1 Introduction

The Sensor Web vision refers to the addition of a middleware layer between sensors and applications [1]. With the exponential growth of the number of sensor devices, Sensor Web systems will interconnect many sensors to several applications and, therefore, deal with heterogeneity and scalability considerations. As information-centric systems, observation delivery is the core feature of Sensor Web systems, which play the role of mediators between observation producers and observation consumers. However, this role is

*This paper is a postprint of a paper submitted to and accepted for publication in IET Wireless Sensor Systems journal and is subject to Institution of Engineering and Technology Copyright. The copy of record is available at IET Digital Library.

made more complex by the increasing number of sensors, which constitute as many sources that produce observations of disparate quality.

Observation consumers (i.e., applications in most cases) often rely on Sensor Webs to retrieve, analyse and create actual *value* from these observations, which are supposed to be of good quality for them. However, according their application domain, these programs do not require the exact same observations. Indeed, even if some of these applications can ask for the same topic (temperature, humidity, etc.), they may actually require different granularity levels (frequency, confidence, etc.). Moreover, application needs may be dynamic and evolve over time. In order to keep serving their initial purpose, Sensor Web systems should provide additional features to be able to deal with observation heterogeneity, assess observation quality and adapt their behaviour in response to these dynamic consumer needs.

Traditionally, Semantic Sensor Webs have been envisioned to address heterogeneity and interoperability issues, allowing the observation representation and the expression of sensor capabilities [2, 3]. However, most of the time, these solutions primarily focus on sensor discovery or sensor abstraction. Surprisingly, only few Sensor Web systems have been designed to adapt their behaviour according to consumer needs. Even if many standards already exist to describe quality (such as ISO 8000 [4] for Data Quality, OGC SWE 2.0 Common Data Model Encoding for Sensor Webs [5] or ISO 19157 [6] for Geographic information quality), the large majority of Sensor Webs usually define their own metrics before delegating Quality of Observation (QoO) management to applications. However, this way of doing causes important interoperability and integration issues as such applications are strongly coupled to a single Sensor Web. Moreover, these applications utterly depends on the quality attributes provided by the underlying solution, with the possibility that some metrics could be missing or poorly implemented.

The specific objective of this survey is to study why and how observation quality is (or should be) addressed in current (and future) Sensor Webs. It makes three original contributions. First, it provides important insights into quality dimensions and proposes to use the QoO notion to deal with information quality within Sensor Webs. Second, it proposes a QoO-oriented review of 29 Sensor Web solutions developed between 2003 and 2016. In order to compare them more easily, we introduce a custom taxonomy to characterise some of their features from a QoO perspective. Finally, it draws four requirements for helping researchers to build adaptive and QoO-aware Sensor Web solutions.

The remainder of this survey is organised as follows. Section 2 introduces the required background, dealing with Sensor Web vision and QoO notion. This section provides important insights into quality dimensions and proposes to use the QoO notion to deal with information quality within Sensor Webs. Section 3 presents the 29 surveyed solutions and describes them from a QoO perspective. Section 4 proposes a taxonomy to describe the surveyed solutions according to three dimensions (*Structure*, *Behaviour*, *System*) that play a role into management of observation quality. Section 5 sums up the lessons learned from this survey by giving important directions for the future development of Sensor Web solutions. Finally, we present existing and relevant work in Section 6 before concluding and giving research perspectives in Section 7.

2 Required Background

2.1 Sensor Web Vision

Back in 2000, the Jet Propulsion Laboratory of NASA first defined a Sensor Web as a "*collection of sensor pods that could be scattered over [...] regions of interest to gather data on spatial and temporal patterns of relatively slowly changing [...] phenomena in those regions*" [7]. In this definition, the term "Web" refers to the intelligent coordination and behaviour of the sensors within a network rather than the World Wide Web (WWW) [8]. Nevertheless, over the years, with the democratisation of sensors, the meaning of "Sensor Web" has evolved slightly to cover the capability to directly retrieve sensor data through Internet-based architectures.

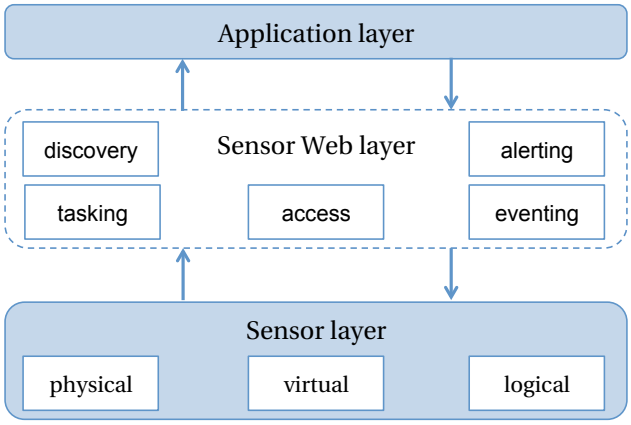

Figure 1: Sensor Web: a middleware layer between sensor and application layers

The Open Geospatial Consortium (OGC)[1] and its Sensor Web Enablement (SWE) working group have defined a Sensor Web as "*an infrastructure that enables an interoperable usage of sensor resources by enabling their discovery, access, tasking, as well as eventing and alerting [...] in a standardised way*" [1]. The OGC SWE working group envisions a Sensor Web as a means to "*bring sensor resources to the Web and make them available to applications*" [1]. Therefore, a Sensor Web can be seen as a middleware layer, filling the gap between sensor and application layers (see Figure 1).

OGC SWE has published several specifications for developers that explain how to implement the Sensor Web vision. The first specification (SWE 1.0) was released in 2007. Since then, standards have been updated and a new generation of SWE specifications (SWE 2.0) is available online[2] since 2011. OGC SWE 2.0 contains several standards (see Figure 2) that address encoding and Web Service specifications. In the following, we briefly describe the most important of them to achieve the Sensor Web vision:

- **SWE Common** defines the common vocabulary used in all others SWE specifications. These definitions encompass data types, parameters and characteristics. Besides, support for "simple quality information" is also mentioned (see also Section 5.1 for a more thorough discussion on Standards).
- **SensorML** defines models and an XML encoding to describe not only sensing process but also higher-level processes used to derive higher-level information from observations. SensorML allows to model a sensor as a process that converts a real phenomenon to an observation.
- **Observations & Measurements (O&M)** specifically address how to model and represent observation, with a clear separation between observations and their features of interest.
- **Sensor Observation Service (SOS)** defines a standard web service interface to retrieve sensor observations. This component enables both the "discovery" and "access" capabilities of Sensor Webs.
- **Sensor Planning Service (SPS)** defines a standard web service interface to allow a user to submit tasks to sensors or request specific observations from them. This component enables the "tasking" capability of Sensor Webs.
- **Sensor Alert Service (SAS)** defines web service interface to publish and subscribe to alerts from sensors. This component enables the "alerting" capability.

[1] `http://www.opengeospatial.org/`
[2] `http://www.opengeospatial.org/ogc/markets-technologies/swe`

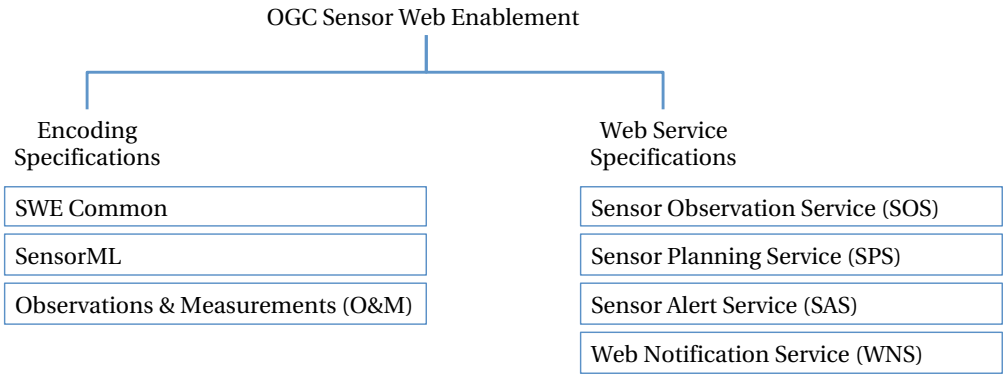

Figure 2: Overview of main OGC Sensor Web Enablement 2.0 specifications

- **Web Notification Service (WNS)** defines a web service interface for asynchronous message delivery ("eventing" capability) that can be enforced within other SWE web services (such as SAS or SPS).

OGC SWE specifications aim at helping researchers, scientists and industrials to design and build infrastructures that provide real-time access to sensor data. Originally, OGC focused on environmental monitoring. As a consequence, first OGC SWE specifications were targeting systems retrieving observations from physical sensors.

Nevertheless, the exponential growth of the Internet of Things (IoT) requires to consider other kind of sensors. Like many researchers, and in particular Perera et al. in [9], we envision physical, virtual and logical sensors. Widely used for environmental monitoring use cases, physical sensors are the most common kind of sensors. They consist of concrete devices that have the ability to generate data by themselves (e.g., a temperature sensor). Virtual sensors are generally Web Services that can be queried through APIs. Contrary to physical sensors, virtual sensors generally do not have a physical presence. For instance, Twitter or Google Maps can be seen as virtual sensors. Finally, logical sensors are sensors that combine information coming from both physical and virtual sensors to produce enriched and more valuable information (e.g., a web service that collects weather data from physical weather stations and displays them on a map retrieved from a virtual sensor).

In this survey, we propose to extend the OGC SWE Sensor Web definition by considering a Sensor Web as *any Web-based system that bridges the gap between any type of sensors (physical, virtual or logical) and higher-level applications*. In the following, we interchangeably use the terms "Sensor Web" and "Sensor Web system". We also use the plural "Sensor Webs" to refer to multiple Sensor Web systems.

Sensor Web systems can be characterised by the kind of observations they provide to applications [10]. Please note that an observation may either be the representation of an observed phenomenon (the temperature of a place, a person that enters a room, etc.) or an event (availability of a new software update for instance). Last but not the least, a same observation may be reported in different ways, including more or less details about the unit of the measure, sensor type, location, etc. In order to estimate the level of abstraction required by end consumers (applications and users) to process and "understand" these observations, taxonomies have been proposed for observation levels. In this work, we reuse the terminology from the "DIKW ladder" [11] proposed by Sheth. In the following, we define these three observation levels from a Sensor Web perspective, from the simplest representation to the richest:

level 1: The first observation level corresponds to unprocessed **Raw Data** directly coming from sensors. Generally, they are encoded in the key/value form and do not contain any additional informa-

tion (e.g., `{sensor_id: 34, value: 20, unit: Celsius, producer: sensor_1}`). Raw Data may contain some contextual details (e.g., the provenance in the example above) but does not require the Sensor Web to retrieve additional Context, on the contrary to the two other observation levels.

level 2: The second observation level corresponds to sensor **Information**. Information is sensor Raw Data that has been processed or enriched with Context (e.g., `{sensor_id: 34, value: 20, unit: Celsius, producer: sensor_1, location: (43.564509,1.468910), accuracy: 0.8}`). To get this kind of observations, Sensor Webs may retrieve additional Context regarding sensors in different ways (via an external database, a context distribution middleware, a sensor API, etc.). In the example above, Context is required in order to associate a physical location to the sensor that produces the measurement. The main distinction between Raw Data and Information is related to the computation and the annotation of quality attributes/metrics to the original observations. On the contrary to Raw Data, Information can be seen as observations that have been "processed" by Sensor Webs.

level 3: The third observation level is reached with the use of semantics. By implementing a semantic annotation approach, Sensor Webs can model domain-specific observations and thus deal with machine-understandable information. We denote by sensor **Knowledge** any semantic-based observation representation (e.g., at `home`, `temperature` is within `comfort` range. This observation can be trusted since it has a `good accuracy`). In order to produce Knowledge, a Sensor Web also requires Context. It also requires a base ontology model to formalise and annotate the different observation fields such as: the geolocalisation of the measure; its quantity and unit; the confidence that one can have in this specific sensor. As a consequence, Knowledge is often produced based on observations coming from level 2 (Information), which already include many contextual attributes.

The reader attention is drawn to the presence of capital letters for terms "Raw Data", "Information", "Knowledge" and "Context" in order to avoid confusion with common sense of these words. Besides, independently of their level, this survey makes use of the generic term "Quality of Observation" (QoO) to denote the degree of observation quality provided to end consumers. We develop more in detail this notion and give examples of QoO metrics in the following section.

2.2 Quality of Observation

In order to define QoO, we build on previous work done in the research field of information quality. In this section, we first recall the definition of Quality of Service (QoS) proposed by the International Telecommunication Union Standardisation sector (ITU-T)[3]. Based on this definition, we explain the interest of considering additional quality dimensions. Since QoO can be seen as a specialisation of information quality applied to sensor systems, most of our findings can be generalised to other information-centric systems.

2.2.1 Quality of Service (QoS)

ITU-T has published several *recommendations* that deal with QoS. In [12], QoS is defined as a set of characteristics and specialisations. A QoS characteristic can be seen as a quality dimension that represents "*some aspect [...] of a system, service or resource that can be identified and quantified*" [13]. These generic characteristics (such as "lifetime") can then be derived into specialisations ("remaining lifetime" or "freshness" for instance) for more appropriate use. Although the definition of QoS encompasses the quality of information, it is not the case in practice. Indeed, the term "Quality of Service" generally refers to packet transportation from source(s) to destination(s) through the network. Therefore, the set of QoS

[3] `http://www.itu.int/en/ITU-T/`

metrics is often restricted to bandwidth, delay, jitter and loss probability (denoted as "network QoS" in Figure 3).

2.2.2 Data Quality (DQ)

Data Quality was first investigated from information systems [14] and customer satisfaction [15] perspectives. In this survey, we consider DQ as the distance between the reported value (i.e., the observation) and the corresponding event (i.e., the physical-occurred phenomenon or the virtual event). DQ is mainly impacted by the sensor device quality (intrinsic quality) and performance of the underlying collection network (especially packet losses and end-to-end delay).

2.2.3 Quality of Context (QoC)

Context plays a major role in Quality of Information (QoI) assessment. 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*" [16]. Like Sanchez in [17], we make the distinction between sensor data and Context. In this survey, we use the term "Raw Data" to refer to raw data coming from sensors and the term "Information" to denote data that has been already processed or enriched with Context (meta-data addition). Just like Service or Information, Context has an associated quality, known as "Quality of Context" (QoC) [18].

2.2.4 Quality of Information (QoI)

in [19], 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*". As a consequence, the definition of QoI attributes is quite generic. Some examples of QoI attributes are latency, reputation and provenance. Table 1 gives five examples of QoI attributes alongside with their commonly accepted definition (not standardised), alongside of some research works that define them. The implementation of these QoI attributes (i.e., the way that applications compute them) should be really concrete and should not vary over time nor from one application to another. On the contrary, QoI assessment should be performed by each application regarding its present needs and the current applicative Context. QoI notion is quite close from the original definition of the QoS, allowing applications to assess more accurately how fit-for-use Information for them is. Since some QoI attributes (such as latency and timeliness) may be impacted by underlying-network performances, QoI and network QoS are two closely linked notions. This should not represent an issue because QoI is not intended to replace network QoS. Instead, QoI and network QoS are two complementary notions that may be used together. For instance, using both network QoS and QoI, an application may better understand if some outdated observations are the result of poor network performances or due to a sensor sampling rate too low. In this case, the notion of "outdated observations" is specific to an application (e.g., only accept observations not more than 2 seconds old) but may be different for another application that retrieve observations from the same Sensor Web. QoI has recently gained attention, in particular within the military domain. Indeed, being able to assess QoI is sometimes critical, especially in tactical sensors where information dissemination is constrained by available resources [20].

2.2.5 Quality of Knowledge (QoK)

Semantic Sensor Web [2] is a category of Sensor Web systems that use semantics to model sensor observations and/or to describe the capabilities of their sensors. Most of the time, these systems use ontologies or another kind of knowledge representation to conceptually represent Knowledge. As a result, several domain-specific ontologies have been defined to model sensor-related thematic fields (such as the

Table 1: Some QoI attributes. For more examples, see [21].

Attribute name	Common definition	Mentioned in
Accuracy	Distance between reported observations and the corresponding phenomenon/event.	[21, 22]
Provenance	Sensor or mechanism that has output the observation.	[21]
Reputation	Publicly held opinion of a sensor or intermediary mechanism.	[21]
Latency	Duration to retrieve an observation (including network transport time).	[21]
Timeliness	Time horizon over which an observation is considered as valid.	[21, 22]

weather or oceanography for instance). This allows semantic queries, inference and high-level reasoning from sensor observations. Ontology-based modelling involves the definition of concepts, relationships and properties for a specific domain. Using semantics for observation modelling corresponds to transforming Raw Data or Information into machine-understandable Knowledge. Within Semantic Sensor Web systems, sensor capabilities are described in a semantic way (what is their type, their sampling rate, their units, etc.) [23]. Since Semantic Sensor Webs consider sensors as abstract observation providers, it enhances their reusability and global system interoperability. Several works have shown that sensor addition and removal (also called sensor *plug-and-play*) were easier within such Sensor Webs [3]. QoK assesses the quality of the ontology-based modelling. Some metrics (such as completeness, coverage and ease to use) have been proposed for Knowledge management systems [24].

We now explain the different relationships between the quality dimensions introduced so far. Network QoS impacts all others quality dimensions as it affects the transportation of observations from sensors to Sensor Webs, and then from Sensor Webs to final consumers. Whether for virtual or physical sensors, these collection and distribution phases introduce additional delay and may affect the intrinsic DQ. Depending on the medium characteristics and the transport protocols used, observations may be prone to other issues such as packet losses. Since Information (level 2) and Knowledge (level 3) are produced from Raw Data (level 1), DQ may impact both QoI and QoK. Besides, QoI largely depends on the quality of the available Context (which may be retrieved or inferred). As a result, QoC also impacts QoI attributes. Finally, since Knowledge is often derived from Information through semantic annotation processes, QoI may have an impact on QoK. Figure (see Figure 3) summarise the different quality dimensions and relationships that we consider in this survey.

This section has introduced different quality dimensions. Since Sensor Webs are observation-centric systems, it is critical that applications may be able to characterise and assess QoO. Based on previous definitions, we propose to use the term "**Quality of Observation**" (QoO) to denote Data Quality, Quality of Information or Quality of Knowledge, according to the application consumption level.

3 Evaluation of existing Sensor Web solutions

3.1 Methodology

The goal of this survey is to carry out an extensive survey of Sensor Web systems from a *Quality of Observation* perspective. By analysing a significant number of solutions, we wanted to have an overview of the most popular QoO-related and adaptation-related mechanisms. To achieve this in a rigorous manner, the following five-step methodology was used:

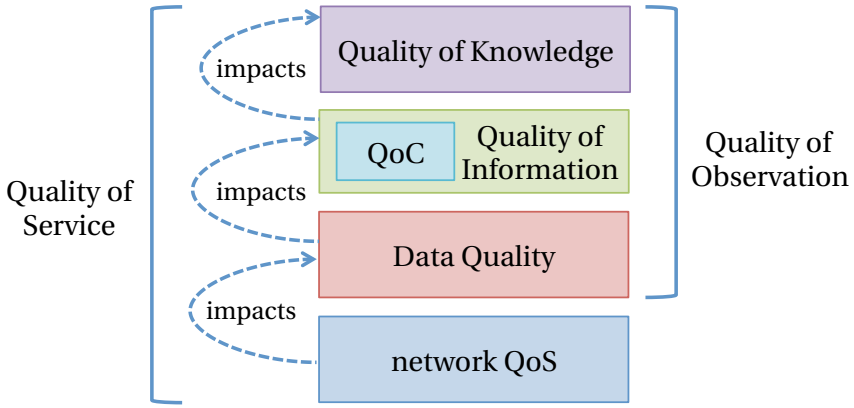

Figure 3: Quality dimensions and relationships considered in this survey

1. Prior to analysing solutions, we selected some of them that had interest in QoO. For our study, a QoO interest was assumed from the moment where the authors of a solution discussed (in a peer-reviewed publication or in the software documentation) about any of the quality dimensions introduced in Section 2.2. As expected, it turned out that a large number of Context-aware sensor middlewares met this criterion. In order to decide if a solution should be integrated into this survey, we then analysed its QoO management (metrics used, mechanisms, adaptation features and originality) and its compliance regarding OGC SWE specifications. Because network QoS has a direct impact on QoO, we also investigated for mechanisms that could provide QoS guarantees and therefore improve overall QoO. We ranked these solutions based on the number of quality metrics and mechanisms found, as well as on the average number of citations of each solution from Google Scholar. We also investigated the Related Work for each selected solution to identify whether a solution was inherited from another one (reuse or extension). We ended up with 29 Sensor Web solutions implemented between 2003 and 2016.
2. The second phase consisted in the extensive analysis of these 29 solutions. In order to have an overview of each solution features, we selected the most complete peer-reviewed research paper that was describing it. In the case of projects or open source solutions, we also browsed documentation and official websites when available. Section 3.2 briefly summarises the main features offered by each Sensor Web solution.
3. After having analysed the different features for the 29 solutions, we wanted to compare them in terms of their "QoO awareness". In the case of our quality-oriented survey, we found that each surveyed Sensor Web solution could be characterised by a few set of top-level characteristics. Therefore, we proposed a custom taxonomy that envisions some *Structural*, *Behavioural* and *System*-related features. Section 4 describes more in details this taxonomy as well as its top-level categories.
4. In order to synthesise our findings, we gathered the 29 surveyed solutions into the Table 2. Each row represents a Sensor Web solution while the columns refer to the different features that it provides (i.e., the different categories of the taxonomy). For a better readability, we use abbreviations that are detailed in Table 3. A dash symbol (-) indicates that the solution does not implement/support a feature or that this feature is non applicable for the Sensor Web considered.
5. Finally, the study of concrete solutions allowed us to identify some important challenges for the design and the development process of a Sensor Web solution. Among them, we chose to present four of them that were regularly cited as open or research challenges. Thus, Section 5 present some lessons

learned related to standards, layer-based architecture, mediation, adaptation and reconfiguration.

3.2 Surveyed solutions

IrisNet [25] is an agent-based Sensor Web solution. It is composed of *Sensing Agents* and *Organizing Agents*. Users can customize the behaviour of *Sensing Agents* by writing custom functions called "senselets". These functions are then applied to the raw sensor feeds coming from sensors in order to report Information to an *Organising Agent*. Computation may also been distributed among several agents, with the use of shared memory pools. Finally, IrisNet provides a service-specific database abstraction for observation storage. For a given service, observations are distributed and retrieved among the participating organising agents in a hierarchical way.

Jiang et al. [26] have proposed a Sensor Web solution for tactical sensor networks. This SOA-inspired solution is based on the notion of *Sensor Web Services*. According to the authors, Sensor Web Services are Semantic Web Services that act as wrappers to communicate with physical sensors. These web services semantically describe their capabilities using the DARPA Agent Markup Language - Service (DAML-S) ontology. Then, users can create custom Information workflows according to their needs. These workflows are performed with a distributed *Peer-to-Peer Fuselet Network*. Within this network, sensor observations are routed through several *signal processing fuselets* that may apply some data fusion algorithms called "fuselets". This Sensor Web solution also supports dynamic reconfiguration by allowing manual workflow modification and fuselet addition.

Ranganathan et al. [27] have proposed a middleware for Context distribution within ubiquitous computing environments. This distributed middleware is built on top of CORBA technology and follows a component-based architecture style. Using agents, this solution aims to collect, process and deliver Context to Context-aware applications. For this solution, the authors envision *Context Providers* as a kind of virtual sensors that only produce Context information. They may use some reasoning or learning mechanisms (e.g., Machine Learning) before sending Context to *Context Synthesizers*. These agents are in charge of collecting and processing the received Context in order to deduce or infer higher-level Context. Within the whole solution, ontologies are used to describe *Context Predicates* according to a <Subject, Verb, Object> triple template.

MiddleWhere [28] is a Sensor Web destined to provide location service from heterogeneous sensing location sources (GPS, RF badge stations, personal computers, etc.). This solution characterises the "quality of location information" with the computation of three metrics ("resolution", "confidence" and "freshness"). To improve quality of location information and better answer to queries, MiddleWhere enables Filtering and Fusion of location information.

MASTAQ [29] solution targets consumers that use sensor data for environmental monitoring. Using a *statistical QoI* Application Programming Interface (API), these consumers can add QoI requirements to their queries. For instance, "standard deviation" and "confidence level" are two metrics used within MASTAQ to specify QoI-based Service Level Agreements (SLAs). When satisfying consumer needs, this Sensor Web takes into account the trade-off between observation accuracy and energy consumption. To dynamically adapt the number of activated sensors, the solution uses a Proportional-Integral-Derivative (PID) controller.

Global Sensor Network (GSN) [30] solution is a component-based distributed solution that relies on the *virtual sensor* abstraction. A virtual sensor may be any kind of sensor (physical, virtual or logical) and hide implementation details for observation retrieval. Virtual sensors may have one or several inputs (including from other virtual sensors) but have exactly one output observation stream. Each virtual sensor supports window processing and continuous queries. To improve scalability, GSN relies on a peer-to-peer architecture combined with a decentralised storage. A publish/subscribe mechanism is employed to retrieve observations. With the use of Transducer Electronic Data Sheet (TEDS) from the IEEE 1451 standard, GSN allows sensor discovery and dynamic sensor plug-and-play.

BIONETS [31] is a bio-inspired Sensor Web for Context distribution. It considers three main entities: *Context-aware Applications*, *Context Sources* and *Context Relays*. The solution is implemented on top of a peer-to-peer REST-based framework, designed to be used in highly dynamic environments (such as ad-hoc or opportunistic computing). Context is semantically annotated and stored within different *Context Relays*, in a distributed way. BIONETS uses a pheromone system (with accumulation, evaporation and spreading mechanisms) to weight and cache observations of interest.

Sensor Web Agent Platform (SWAP) [32] is a Sensor Web based on the OGC SWE standards. This framework addresses the lack of semantics of the OGC SWE solution, aiming to facilitate the development of Sensor Web applications. SWAP still exposes OGC SWE services as non-agent resources. However, compared to the OGC SWE solution, SWAP combines Service-Oriented Architectures (SOA) and Multi-Agent Systems (MAS) paradigms. Ontologies are used at *conceptual level* to describe observations and at *technical level* to describe agent capabilities. This semantic-based representation allows agents to process and reason about observations retrieved from *sensor agents*.

SenseWeb [33] is a Sensor Web created by Microsoft. This proposal envisions the "Shared Sensing" paradigm. Within this solution, observations are uploaded and stored into SenseDB, a sensor-streaming database. Applications can submit queries either to the central *Coordinator* (Raw Data level) or Transformers (Information level). To insure scalability, observations are only retrieved on demand according to application queries and are reused whenever it is possible. Queries may contain network QoS and QoI requirements (with tolerance). In order to optimize sensor selection, the coordinator can learn their characteristics at runtime. Please note that this solution does not comply with the OGC SWE standards.

Bouillet et al. [34] have proposed a semantic-based Sensor Web to cope with heterogeneous sensor networks. Their solution relies on *Processing Elements* (PEs). A PE is a reusable component that can take several observation inputs, process them according to defined rules and outputs a new stream. Each PE is semantically described with the use of ontologies to enable composition and chaining. When a user request arrives, the system performs dynamic PE selection and composition to build a data flow pipeline able to output the desired end results. This pipeline may contain a PE source, none or several intermediary PEs and a PE sink.

AcoMS [35] is a Sensor Web for Context distribution. It supports queries with QoI requirements by considering "uncertainty", "frequency" and "accuracy" metrics. This solution complies with the Autonomic Computing paradigm by implementing several self-* features such as self-configuration, self-reconfiguration and self-healing. Regarding sensors, the jointly use of TEDS from the IEEE 1451 standard and semantics allows dynamic sensor discovery and composition.

SEAMONSTER [36] is a concrete Sensor Web deployment for glacier and watershed monitoring in Alaska. This solution is based on a *Multi-agent Architecture for Coordinated, Responsive Observations* platform. The main purpose of SEAMONSTER is to meet different mission goals (i.e., application requests) while optimizing available resources in highly dynamic environments. The system adaptation is continuously performed by agents (for local adaptation) and server-based agents (for mission objectives).

Wieland et al. [37] have modified an existing Sensor Web to support observation uncertainty. Their solution allows Context-aware applications to specify Quality of Context (QoC) requirements. Within this solution, sensors are in charge to annotate Context (when possible) with additional meta-data such as "reliability" or "resolution" attributes for later reasoning. Using Business Process Execution Language (BPEL) and previous metrics, users may define some Context-aware workflows in order to process and reduce uncertainty of Context (e.g., with Filtering or Fusing).

Pathan et al. [38] have combined Sensor Web vision with the Autonomic Computing paradigm. The solution integrates a Monitor, Analyse, Plan, Execute and Knowledge (MAPE-K) adaptation loop. Adaptation and reconfiguration are triggered according to the events collected from the underlying sensor network. Within this solution, sensors capabilities and observations are semantically described using the W3C Semantic Sensor Network (SSN) ontology. The implementation is based on Service-Oriented Architecture (SOA) and an Enterprise Service Bus (ESB) that allows sensor plug-and-play and self-(re)configuration according to application scenarios and Context.

CAPPUCINO [39] Sensor Web enables Context-awareness for Web Services within ubiquitous environments. Implemented following a Service Component Architecture (SCA), it is composed of three main components: a *MAPE-K autonomic control loop*, an *execution kernel* and a *Context-aware module*. This Context-aware module abstracts sensors as reusable *Context nodes* able to collect Context. These nodes can be chained and customised with custom *Context operators* (e.g., Fusion or Filtering functions) in order to generate *Context reports*. These reports feed the MAPE-K autonomic control loop and may be used to trigger dynamic Web Services adaptation and reconfiguration.

OGC SWE [1] is a standardised Sensor Web architecture supported by the Open Geospatial Consortium (OGC). It provides sensor data encodings and specifications for Web Service interfaces in order to build standardised Sensor Web solutions. According to the OGC SWE, a Sensor Web should "enable an interoperable usage of sensor resources" [1] with adequate Web Services to perform basic operations (discovery, access, tasking, alerting and eventing). This standard does not provide any layer-specific mechanisms (such as Context annotation or QoI computation for instance). Instead, it is up to researchers to extend data encodings with the desired quality attributes. Whether commercial or academic, a large number of solutions have adopted the OGC SWE standards. Most of the time, the resulting proposals extend the reference architecture with new features such as semantics or dynamic adaptation. Even if the last specification dates from 2011 (OGC SWE 2.0), it currently remains the only standardised proposal for Sensor Web systems.

Teixera et al. [40] have proposed a Service-Oriented middleware for the Internet of Things (IoT). Its main focus is to ensure interoperability and scalability while considering a large number of underlying sensors. The authors use semantics to describe observations (*Domain Ontology*), sensors (*Device Ontology*) and possible QoO adaptation mechanisms (*Estimation Ontology*). These mechanisms are estimation models (e.g., "linear interpolation") that may be used to design *dataflows*. Once selected by the mean of an approximately optimal composition mechanism, a dataflow may be executed to meet user requests.

Semantic Network Monitoring, Analysis and Control (SNoMAC) [41] is a Sensor Web for network monitoring and control. This solution relies on the NetCore ontology, which may be extended with ontologies specified within *network adapters*. This design allows the use of SNoMAC over various networks, which use different communication protocols (UPnP, TR-069, etc.). The final ontology used within SNoMAC considers three entities: *Networks*, *Nodes* and *Links*. A Node and a Link may have an associated *State* while only Nodes expose *Actions* which they can locally perform. To demonstrate the SNoMAC's extensibility, authors present an integration example with the SIXTH sensor middleware.

The **Linked Stream Middleware (LSM)** [42] is a solution which enables Linked Data [43] for observation streams within Sensor Webs. The Linked Data paradigm consists in publishing and organizing data from different sources through the Web. Even if this Sensor Web does not provide any adaptation nor observation quality features, it allows access to various sensor data sources through different wrappers. These wrappers allow to semantically annotate observations coming from physical sensors (with *physical wrappers*), other systems including other Sensor Webs (with *mediate wrappers*) as well as databases (with *LD wrappers*). The semantic annotation of observations is achieved using the W3C SSN ontology.

Kali2Much [44] is a Sensor Web for adaptive collection and distribution of Context. Context-aware applications may submit queries to Kali2Much by semantically specifying *what* they need as Context. Like many Context distribution systems, this solution follows a Data flow architecture. Each query triggers a reconfiguration of the Sensor Web in order to create a Context flow pipeline composed of *KaliSensors*, *Context Collectors* and *Context Transformers*. The behaviour of Context Collectors can be customised with consumer rules. The main goal of *Context Transformers* is to perform unit conversion and to adapt representation according to consumer needs.

MobIoT [45] is a Sensor Web to cope with challenges of mobile IoT. MobIoT revisits Service-Oriented Architecture by proposing a "Thing-Based" SOA relying on an environment-aware middleware. This solution addresses unknown topology and scalability considerations with internal probabilistic mechanisms (for both providers registration and consumers look-up). It also uses semantics to cope with sensor and observation heterogeneity. To query MobIoT, consumers may submit *Thing-based queries* charac-

terised by a *Physical concept*, a *Unit* and a *Location*. Lastly, consumers may extend a concept ontology by specifying their own "fusion functions".

INCOME [46] is a QoC-based Sensor Web for Context distribution. It allows consumers to express SLAs containing their QoC needs. The distribution of Context is then performed according to the consumer needs, in a distributed way between Context producers. INCOME framework can be divided into two main components: *muContext* for SLAs specification and filter creation; *muDEBS* for the implementation and the routing of Context among brokers. Within muContext, some components called *Context processing capsules* may perform high-level processing and reasoning tasks on Context (Fusion, Aggregation, etc.).

DQS Cloud [47] is a Cloud-based Sensor Web for sensor services. The authors mention the "data quality" (DQ) support as a core feature of their system. In order to be consistent with the previous definitions on quality dimensions (see Section 2.2), we rather consider that this solution provides "Quality of Information" support. DQS Cloud performs sensor selection based on both feed *content* and feed *quality*. DQS Cloud can plan *feed processing workflows* either on gateways or Cloud servers to optimize both bandwidth and battery consumption. By assuming control on its observation providers, DQS Cloud is able to provide recovery in case of QoI degradation or sensor failures.

CASSARAM [48] is a Context-aware tool to enable the "Sensing as a Service" paradigm. Inspired from the Cloud Computing paradigm, this service model consists in selecting an optimal subset of sensors based on some QoI attributes (such as "availability" or "accuracy" for instance). CASSARAM extends the W3C SSN ontology to 1) semantically describe sensor capabilities and 2) characterize observations with Context annotation. The authors demonstrate that CASSARAM can be easily added to an existing Sensor Web by presenting an integration example with the GSN solution. It should be noted that CASSARAM tool only provides inference on sensor capabilities and do not provide additional common QoO mechanisms (like observation Filtering or Caching for instance).

SIXTH [49] is an OSGi-compliant[4] Sensor Web solution. The solution uses several software design patterns and proposes several object-oriented models to represent sensors, queries, observations, etc. These models are enough generic to be customised and extended, as shown by the evaluation section and the numerous use cases considered. The observation distribution is done through a *Data Broker* component, which implements the publisher/subscriber design pattern. Regarding adaptation, SIXTH supports runtime reconfiguration with the definition of *Tasking Messages*. Even if this solution does not comply with OGC SWE standards, it falls into the Sensor Web vision by providing sensor abstraction and extensibility while enabling runtime reconfiguration feature.

OpenIoT [50] is an open source Sensor Web for the IoT. It allows on-demand access to IoT services through a Cloud-based architecture while enabling the Linked Data paradigm. OpenIoT is based on X-GSN, an extended version of the GSN solution. Compared to GSN, X-GSN adds support for the semantic annotation of observations according to a domain-specific ontology. OpenIoT envisions observation collection from mobile sensors through *CUPUS*, a quality-aware publish/subscribe middleware. This CUPUS middleware supports both network QoS and QoI requirements and may provide observation preprocessing to reduce the battery consumption on mobile devices. This can be achieved by performing observation Fusion or observation Filtering with a sliding window for instance.

Kibria et al. [51] envisions the "Web of Objects" (WoO) and provides a three-tier Sensor Web architecture. This architecture comprises *Virtual Objects*, *Composite Virtual Objects* and *Service* levels. This solution semantically describes its services, resources and devices with custom ontologies. It also considers both *Sensors* and *Actuators* as virtual objects that can be reused to build composite virtual objects. The whole adaptation process is Context-aware and involves reasoning and inference. When needed, reconfiguration is triggered and achieved with dynamic service composition.

CityPulse [52] project is a framework for providing large-scale stream processing solutions to Smart City applications. This Sensor Web offers all QoO mechanisms that we consider in this survey, whether for

[4] `https://www.osgi.org`

layer-specific or common mechanisms. Regarding semantics, CityPulse extends the W3C SSN ontology with the "Quality Ontology" and the "Stream Annotation Ontology" among others. These ontologies have been specifically defined to address QoI characterisation and QoI assessment for observation streams. To compute QoI, CityPulse uses custom-defined *collection point-related Key Performance Indicators* (KPIs). Many reusable tools and components (such as the *CityPulse QoI Explorer* for instance) have been developed for this project and are available online at a public repository[5].

FAPFEA [53] is a proactive solution that aims to increase availability of Web Service in the Sensor Web field. This proposal follows the OGC SWE standards and architecture. The authors have developed a special virtual sensor, called *Proximity Sensor Service*, used to predict next response times and estimate replication requirements. These estimations are performed using Fuzzy Logic in order to reduce decision uncertainty. This mechanism triggers system reconfiguration. In case of failure or a poor response time, a given Web Service is replicated on another server and the previous instance is decommissioned.

4 Taxonomy of surveyed solutions

A Sensor Web is primarily a software system. As such, multiple models or architectural views may be used to represent it. To draw a parallel, the Unified Modeling Language (UML)[6] is often used to conceptually represent the structure and behaviour of software prior to its development, with class diagrams, sequence diagrams and state-chart diagrams for instance. In this survey, we adopt a reverse-engineering approach to describe the surveyed solutions (apart from the sensors and applications connected to it) according to some of their QoO-related features.

To achieve this, we created a simple custom taxonomy that distinguishes three top-level categories. The **Structure** category refers mainly to the development and the deployment phases of a Sensor Web. It describes some immutable characteristics that are not likely to change over time like the observation encodings, the main software components and their interfaces, etc. The **Behaviour** category refers to the internal behaviour of a Sensor Web during the running phase. It relates to some dynamic mechanisms that can be deployed or enforced in order to better meet consumer needs. For instance, after a user request, a Sensor Web may start filter out some observations or cache some observation values. It can also dynamically enable or disable semantic annotation according to the observation level asked by a given consumer. Finally, a Sensor Web solution may also have some **System**-related properties that depend or impact both their Structure and Behaviour.

Figure 4 presents the taxonomy we used to describe and compare the 29 surveyed solutions between them. Please note that this taxonomy is not exhaustive and may be not suitable for describing other Sensor Web systems. The only purpose of this taxonomy is to compare the surveyed solutions in order to identify current trends and future requirements for QoO-aware Sensor Webs. The category names and properties have been kept quite generic on purpose in order to limit the taxonomy complexity. In the following, we detail each top-level category and the different properties associated to them, explaining why they are important to consider regarding QoO.

4.1 Structural considerations

The way a Sensor Web has been designed and deployed relates to its *Structure*. This latter is unlikely to change over time and therefore can be qualified as immutable. Being able to associate one or more architecture styles to a Sensor Web allows to conceptually visualise it alongside its main components (e.g., client/server) while hiding implementation details. The fact that a solution complies with some standards may also give additional information on its software components or observation encodings. For instance,

[5] `https://github.com/CityPulse`
[6] `http://www.uml.org/`

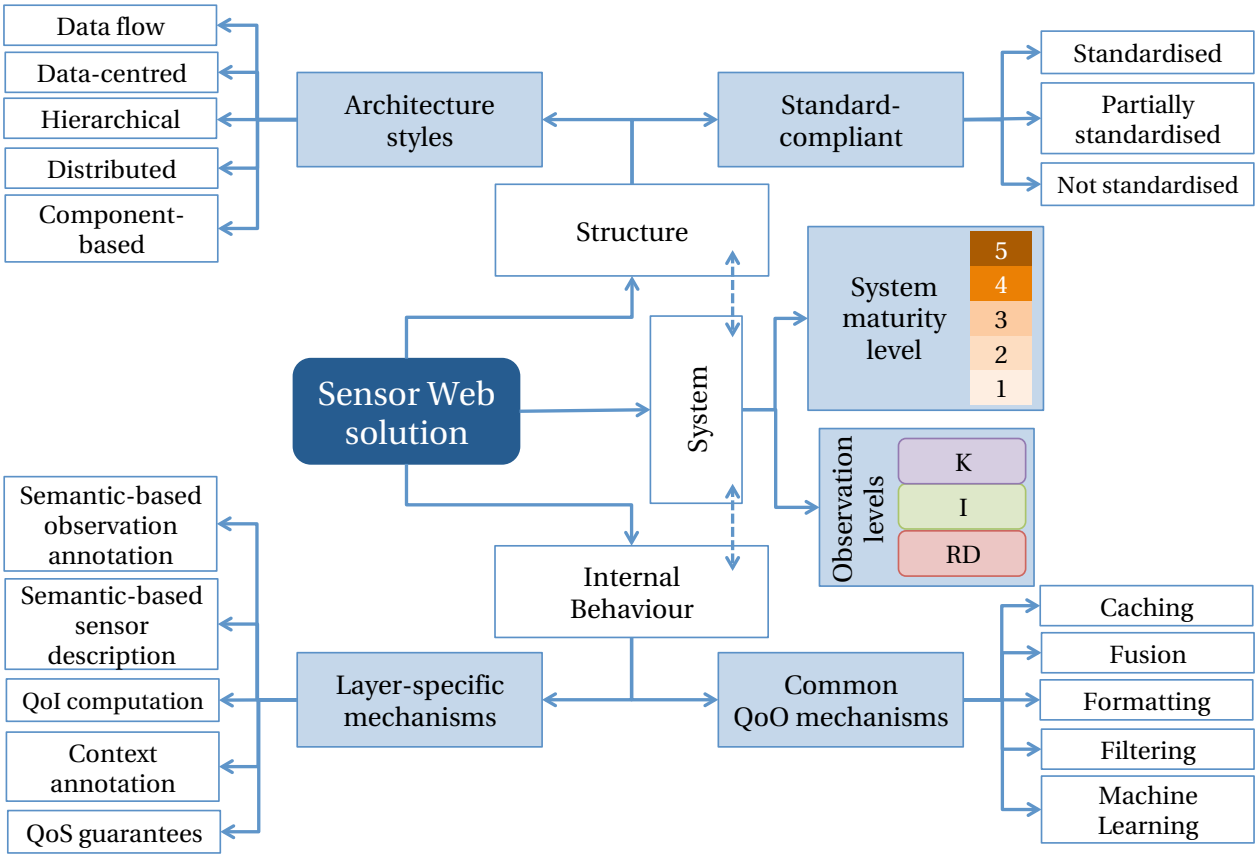

Figure 4: Custom taxonomy used to describe the surveyed Sensor Web solutions (RD: Raw Data, I: Information, K: Knowledge). System-related properties may affect or depend on both Structural and Behavioural properties.

due to specifications, a Sensor Web that complies with OGC SWE standards is generally a distributed system composed of many web services that exchange O&M-encoded observations.

4.1.1 Architecture styles

according to the architecture style of a Sensor Web, the nature of the quality mechanisms (specifically regarding network QoS and QoO) and the ways to enforce them may differ. Besides, the entities in charge of quality management may be different as well. For instance, a data-centred solution that uses a shared data store may have a global view of all received observations. In order to improve QoO, this kind of Sensor Web can run batch jobs that will fetch, transform and update the observations saved. On the contrary, a component-based Sensor Web solution may rather have several local views that correspond to the vision of each of its components. These components may be asked to locally perform some tasks before reporting the results of their work to a more central entity such as a manager. In the end, this manager may only have a partial and aggregated vision of the received observations. To improve QoO, this kind of Sensor Web will rather enable composition and distributed cooperation between its components in order

to meet the asked QoO level.

This survey distinguishes five main software architecture styles, which may overlap in some cases. Our goal here is not to provide an exhaustive list but understand what are the most popular architectural trends when it comes to Sensor Webs instead:

- **Data flow** architectures generally consist in a succession of several transformations applied on observations. Within such Sensor Webs, we witness the creation of "observation pipelines" where the content, value or encoding of observations evolve as they go through the different modules of the system. Each module may take many inputs, perform a specific operation (Fusion, Filtering, etc.) and output observations to another module or to its final consumer(s).
- **Data-centred** architectures organise themselves around observations. This kind of Sensor Webs is characterised by a shared data store (e.g., database) which maintain the current state of sensors, observations, alerts, etc. Other services may be designed to access this data store and perform some computations, before pushing back the results. Finally, we also classify the Sensor Webs that perform Context distribution as data-centred architectures.
- **Hierarchical** architectures refer to a system where tasks may be decomposed into subroutines. For instance, a master component may divide a task into smaller ones before sending them to its slaves for treatment. Finally, the master may receive the final results and perform selection, aggregation, etc. Figuratively, a hierarchical architecture may also denote a layer-based Sensor Web where each layer has a given responsibility and is served by the layer below it (just like in the conceptual OSI model where TCP and UDP layers can rely on the functionalities provided by the underlying IP layer for instance).
- **Distributed** architectures are composed of several components located on different physical entities. The different components work together and communicate over a communication network (middleware, message broker, Internet, etc.) to collectively achieve a specific objective. Please note that we decided to include Service-Oriented Architectures (SOA) and Cloud-based Sensor Webs within this specific category.
- **Component-based** architectures rely on the definition on well-defined and reusable entities called components. For instance, Web Services, actors and agents are some popular component implementations. Each component takes care of a sub-problem, providing a particular and well-identified service to others. Component-based Sensor Webs enable reusability and composition of existing components, which are generally selfish entities.

4.1.2 Compliance with standards

developing a new Sensor Web solution from scratch can be difficult. Instead of reinventing the wheel, researchers and developers should reuse standards. This will have for effect to reduce implementation efforts while improving the future extensibility of the whole system. Indeed, standards enable interoperability at various levels. They may help to integrate many information sources to a Sensor Web or to connect many Sensor Webs between them, allowing mutual observation sharing. In order to achieve this horizontal vision, standards have been defined for software architectures, encodings, quality attributes and may define precisely how to integrate various sensors, what quality attributes to use and when, how to compute them, etc. We give more details about standards, their use and adoption in Section 5.1.

This specific taxonomy category aims to evaluate the standard compliance for each Sensor Web system with regard to software architectures, encodings and the definition of quality attributes. From the surveyed solutions, we identify three major trends regarding the use of standards within Sensor Webs (see column (7) of Table 2):

- **Standardised** (e.g., OGC SWE [1])
  Solutions that are mainly based on SWE standards. By using well-known interfaces and data encoding, they enable interoperable access to their sensors.
- **Partially standardised** (e.g., SIXTH [49], GSN [30])
  Solutions that do not comply with all SWE specifications. However, they may use some standards for their observation encodings (such as SWE SensorML or the SSN ontology developed by the W3C[7]) or their software architectural design (such as those defined by OSGi[8]).
- **Non-standardised** (e.g., SenseWeb [33])
  All solutions that cannot be categorised into the two previous Sensor Web types. Often, these solutions define ad-hoc proprietary interfaces and data encodings, only using popular standards as implementation choices (e.g., XML, JSON, etc.). However, none of these standards can be used to address QoO concerns within Sensor Web systems. Therefore, we qualify the solutions that only use such standards as "non-standardised".

4.2 Behavioural considerations

As said before, a Sensor Web solution may adapt its internal *Behaviour* to provide adaptation and reconfiguration features. In the case where end consumers can express some QoO constraints, the system may be required to specifically process observations according to consumer needs, leading to the creation of dynamic observation pipelines with distinct quality levels. We call "QoO mechanism" any reusable piece of software that can be applied to one or more observations, resulting in a transformation of these observations. These mechanisms, comparable to mathematical functions, can be chained to obtain more complex treatments on observations.

We distinguish two different kinds of mechanisms. On the one hand, we denote by layer-specific mechanisms the ones that are highly tied to the asked observation level. For instance, "Semantic-based observation annotation" requires the use of semantics and an ontology model and, therefore, is characteristic of observations produced by the Knowledge layer. Almost all the layer-specific mechanisms (excluding "QoS guarantees") are some functions that perform semantic enrichment. On the other hand, we denote by QoO common mechanisms more generic mechanisms destined to improve (if possible) one or more aspect of QoO. For instance, common mechanisms may consist in modifying, converting or dropping specific incoming observations to improve the overall QoO of delivered observations to each of final consumers.

We chose to describe these mechanisms from a high-level perspective to not be subject to additional Structure-related considerations. Indeed, a Sensor Web solution may provide one or many of the following QoO mechanisms, independently of the fact that it uses standards for its observation encodings (or not):

4.2.1 Layer-specific mechanisms

according to the number of observation levels that the system can provide, some layer-specific mechanisms may be enforced. We qualify these mechanisms as layer-specific as each of them may operate on a single type of observation level or may require particular material that may not be available at other observation levels. For instance, QoS guarantees can be enforced to better retrieve observations from sensors. Since we assumed that sensors only output Raw Data, QoS guarantees is therefore a layer-specific mechanism. In the same way, Context annotation and QoI computation may be used to produce Information from Raw Data. Finally, semantics and ontology-based modelling requires from the Sensor Web to already have a base ontology model at Knowledge level. From the surveyed solutions, we selected the five most popular of them. Almost all of them (excluding "QoS guarantees") are related to observation enrichment. In the following, we explain how they may contribute on QoO assessment or improvement:

[7] `http://www.w3.org/`
[8] `http://www.osgi.org/`

- **Semantic-based observation annotation** is used in Semantic Sensor Webs, generally to annotate observation with domain-specific ontologies. This mechanism is characteristic of the *Knowledge* observation level. By allowing to annotate observations with already existing concepts, this mechanism is a key enabler for QoO assessment, especially when Sensor Web are required to reason about some quality attributes (e.g., timeliness is impacted by end-to-end network QoS latency so outdated observations may have several causes). Sensor Webs may then use this Knowledge to take appropriate decisions and adapt or reconfigure themselves.
- **Semantic-based sensor description** consists in the maintenance of a registry where sensors are semantically described. This mechanism allows Sensor Webs to have an overview of the capabilities of the underlying sensors. For instance, a physical sensor may have an API endpoint to query its battery level, change its sampling rate, etc. In case of a sudden QoO degradation, a Sensor Web may ask this sensor to temporarily increase its sensing rate. More generally, semantic-based sensor description is helpful for adaptable Sensor Webs to plan and execute remedies.
- **Context annotation** consists in enriching Raw Data with Context (see the previous QoC definition introduced in Section 2.2). Sensor Webs that cannot retrieve additional Context (or that retrieve Context of low quality) may produce Information observations of lower quality and, by extension, low-QoO Knowledge. Indeed, these two observation levels require some details that are generally not included within Raw Data observations (such as sensor capabilities or geographical location for instance). In terms of QoO attributes, Context annotation may mainly improve accuracy and completeness for Information/Knowledge observations.
- **QoI computation** is achieved by the analysis and processing of annotated Context to derive new metrics (such as *freshness* or *trust*). Again, this mechanism helps Sensor Webs during QoO assessment or observation delivery. Indeed, a Sensor Web may pre-filter some observations if they do not meet minimum QoO needs for a given observation consumer. In the end, this leads to globally improve QoO for the observations effectively received by this consumer.
- **QoS guarantees** denote a support for network QoS requirements (regarding *jitter*, *bandwidth*, etc.) by the system. A minimum network QoS is often needed to not introduce additional latency during the transport of observations. The lack of QoS guarantees may directly impact QoO and cause additional latency and packet losses.

We describe more in detail these layer-specific mechanisms in Section 5.2. In particular, we discuss which observation levels are responsible to enforce them, when applicable.

4.2.2 Common QoO mechanisms

unlike layer-specific mechanisms, common QoO mechanisms are not tied to a specific observation level. Therefore, a Sensor Web may apply them on any observation kind (subject to adaptation to the observation encodings). From our survey, we have found the following mechanisms to be the most popular ones:

- **Caching** mechanism allows Sensor Webs to consume fewer resources and to respond faster to queries, even if sensors are disconnected. To cope with that issue, different kinds of caching (distributed/centralised) and different caching strategies (proactive/reactive) may be used. Of course, one of the greatest challenges is to continuously cache relevant and up-to-date sensor observations. It should be noted that caching may be challenging within distributed systems where observation consistency cannot be achieved without compromises regarding availability, tolerance to network partitions and latency (see the CAP theorem [54] formulated by Brewer for more details).
- **Fusion** mechanism is generally used to reduce stochastic errors of observations [37, 55] in order to reduce observation uncertainty. One of the greatest challenges is to fuse observations without

any losses. Nevertheless, some use cases (such as environmental monitoring for instance) are less sensitive to these losses as observation consumers often only care about orders of magnitude.

- **Formatting** refers to the process of adapting the presentation of the observations to customer needs. Even if formatting is not supposed to modify QoO, it can make it more or less understandable to final consumers (e.g., temperature conversion into Fahrenheit or Celsius degrees).
- **Filtering** mechanism aims to ensure that observations comply with some thresholds. This adaptation mechanism is particularly popular within Context distribution systems [46]. It is generally implemented with Publish/Subscribe delivery channels for instance.
- **Machine Learning** is a mechanism that may be used when some observations are not available but still predictable. In this case, largely depending on the Machine Learning algorithm, the system should have sufficient historical data to make the most accurate predictions possible.

4.3 System-related considerations

Depending on the offered layer-specific mechanisms, a Sensor Web may provide several observation levels (e.g., Knowledge thanks to Semantic-based observation annotation). However, this feature is also depending on the observation encodings (or any standard) used, which is a Structure-related choice made at design phase. Since this property is both conditioned by some *Structural* and *Behavioural* properties, we included it into the *System* category. Another cross-category property is the ability of a Sensor Web to adapt and reconfigure itself given its "System maturity level". Indeed, this ability depends both on available QoO mechanisms and on the software components that will enforce these mechanisms (Where should it be enforced?, Do we need synchronous or asynchronous deployment?, etc.). In the following, we describe these two *System*-related considerations:

4.3.1 Observation levels supported

based on previous definitions from Section 2, this survey only considers **Raw Data**, **Information** and **Knowledge** as observation levels. To determine the observation level(s) that a Sensor Web supports, we focus on the encodings and characteristics of observations delivered to final consumers (see column (4) of Table 2).

4.3.2 System maturity level

to quantify how autonomous a Sensor Web is (see column (5) of Table 2), we reuse the work of IBM on the five levels of autonomic maturity [56]. In the following, we describe them from a Sensor Web perspective:

level 1: at this level, no customisation is feasible by consumers. The entire behaviour of the system is hard-coded by developers at design phase. Developers manually perform system monitoring and update the different elements and components accordingly. We denote as **basic** the behaviour of such Sensor Webs.

level 2: at this level, adaptation is based on predefined rules written by applications or developers. These rules are simple (*if a then b* for instance) and are generally written by a skilled person. Such Sensor Webs have **managed** behaviour.

level 3: **predictive** behaviour is reached with the implementation of reasoning processes in some components of the Sensor Web. These processes may consist in complex treatments (learning, fusion, etc.) but they must not take into account any macroscopic goal. At this maturity level, components provide simple adaptation and are generally selfish entities. For this survey, we also consider as predictive systems the Sensor Webs that provide sensor plug-and-play.

level 4: **adaptive** behaviour is characterised by the definition of Service Level Agreements (SLAs). SLAs mostly correspond to the definition of consumer profiles with specific QoO needs. At this maturity level, local components take into account SLAs to self-adapt their behaviour. Moreover, the whole system may also support re-configuration with dynamic selection and composition of sensor sources.

level 5: the last maturity level is the **autonomic** one. We assume that a Sensor Web is fully autonomic when its behaviour is driven by the expression of business rules coming from end consumers and when it implements a continuous adaptation control loop (such as the IBM MAPE-K one [57]). Such a system automatically derives appropriate SLAs from these rules and distributes them to its different entities. Then, these autonomic entities accordingly adapt their behaviour and collectively fulfil application needs.

All together, these characteristics are helpful in order to compare the surveyed solutions from a QoO perspective. Beyond Sensor Web characterisation, this taxonomy highlights the strong dependencies that can exist between design, development and deployment phases. Like many other software requirements (especially non-functional ones such as scalability, security, etc.), we are convinced that QoO support should be considered at all phases of the creation of a new Sensor Web. Under this approach, the next Section explains the lessons learned from the surveyed solutions and gives directions for adding QoO support to future ones.

5 Requirements for future solutions

The review of concrete solutions has allowed us to identify some key requirements that future Sensor Web should satisfy. In particular, we detail four recurrent requirements that we observed as essential to build an adaptive and QoO-aware solution. Each of them is related to one or many categories of the taxonomy previously introduced in Section 4 that we indicate between parenthesis:

R1-archi. A Sensor Web solution should be interoperable, evolutive and extensible (Structure).

R2-levels. A Sensor Web solution should be able to deliver many observation levels to end consumers (System).

R3-needs. A Sensor Web solution should act as a mediator between sensor capabilities and consumer needs with specific QoO requirements (System and Behaviour).

R4-adapt. A Sensor Web solution should support adaptation and reconfiguration to cope with failures, sensor variability, and evolution of consumer needs (System and Behaviour).

5.1 Standards

Well-adopted standards generally provides guidelines in order to build interoperable Sensor Web solutions and meet the requirement **R1-archi**. As previously mentioned, the OGC SWE has been one of the first to propose a definition for Sensor Web systems, which were primarily used for environmental monitoring at the time. As new paradigms and new systems emerged, the definition for Sensor Webs has evolved. Nowadays, the Internet of Things (IoT) [58, 59, 60] implies to also consider logical and virtual sensors. Just as Sensor Webs, the IoT has seen the proposition of many standardisation attempts like the IoT-A framework [61]. Whether it is for Sensor Webs or the IoT, no standard has become a global trend these days. This statement can be easily explained by two main factors. First, many frameworks try to be the most generic possible to encompass several use cases while many sensor-based systems only target a specific one (e.g., Smart Cities). Second, they are often too complicated to put in practice or require a significant learning phase from developers. As a direct consequence, many of the new sensor-based solutions developed do not comply with any OGC SWE standard. From the reviewed solutions, only three solutions comply with OGC SWE standards. In addition to OGC SWE [1], we can cite SWAP [32] and FAPFEA [53] solutions. These two solutions have reused some available implementations from the

52°North Sensor Web Community[9] in order to speed up the development process of their "standardised" Sensor Web solution.

It should be noted that OGC SWE standards define conceptual models rather than ontologies. However, these models can easily be used as ground concepts to develop ontologies. Some individual research efforts [62, 63] have demonstrated the feasibility to design a Semantic Sensor Web with Semantic Sensor Observation Service (SemSOS). Besides, Semantic Sensor Webs have stimulated the proposal of many ontologies, creating a need for standardisation. Between 2009 and 2011, the Semantic Sensor Network Incubator Group[10] of the W3C initiated a standardisation process. They reviewed 17 sensors and observations ontologies [64], making a distinction between ontologies whose aim is to model domain-specific Knowledge (denoted as observation-centric) and others that describe sensor capabilities (denoted as sensor-centric). After having identified the most relevant concepts, this group developed the Semantic Sensor Network (SSN) ontology [65]. More recently, the W3C has announced the development of a new SSN ontology version[11] in partnership with the OGC. This new release will allow better alignment with OGC SWE core concepts (especially regarding the *observation* concept) and will support a wide range of applications and modern IoT-related use cases.

The use of semantics, and in particular ontologies, is an excellent manner of achieving interoperability between systems. Indeed, due to their structure and definition, ontologies enable the reuse of domain-specific Knowledge. Moreover, they can easily be reused, enriched or extended. Our survey shows that 19 out of 29 solutions use ontologies to either annotate their observations or describe sensor capabilities, confirming a "semantic trend" and an interoperability willpower. CASSARAM [48] and OpenIoT [50] solutions have both extended the W3C SSN ontology to enable Context annotation and observation collection from virtual sensors, respectively. As a last example, SWAP [32] has reused the Semantic Web for Earth and Environmental Terminology (SWEET) ontology [66], developed by NASA for Earth and environmental observation modelling.

Returning to the issue of QoO, some standards have been published to unify the definition and the meaning of quality attributes. ISO 8000 [4] is a global standard for Data Quality. However, it is not suitable for Sensor Webs as it assumes that Data is always business-related (information about clients, products, operations, etc.). Therefore, it has been rarely used for sensors. Common Data Model Encoding is the OGC standard for observation encodings in SWE 2.0 [5]. It mentions that it is possible to annotate a measurement value with *"any scalar data component, in the form of another scalar or range value"* [5]. However, this standard does not explain which attributes should be used or how to compute these quality measures. Lastly, the ISO 19157 standard [6] aims to define attributes and procedures for Geographic information quality. Even if the use of this standard could be somehow suitable for some Sensor Webs, the fact that ISO standards are not freely available restricts their adoption.

Due to unsuitable/paid standards or because of a complicated learning phase, the large majority of Sensor Webs are not standardised and define their own metrics before delegating QoO management to applications. The best illustration of this trend is the EU FP7 CityPulse project that has defined a list of Observation Point KPIs (actually quality attributes) online[12] to explicit the considered metrics used to assess QoO. For many non-standardised Sensor Webs, quality attributes are not destined to be exchanged with other systems: these custom metrics only serve to ensure local QoO assessment of the observations received from the sensors within their organisation area (vertical observation silos). This reduces interoperability and integration possibilities as an application developed to run on top of a given Sensor Web may not be able to run on some other ones.

[9] `http://52north.org/communities/sensorweb/`
[10] `http://www.w3.org/2005/Incubator/ssn/`
[11] `http://w3c.github.io/sdw/ssn/`
[12] `http://iot.ee.surrey.ac.uk:8080/eval.html`

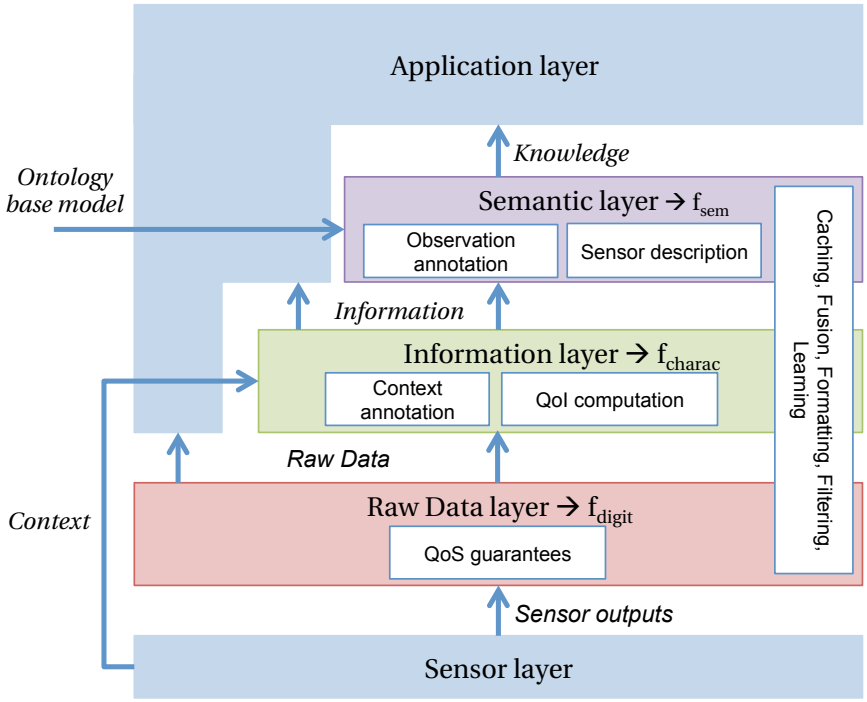

Figure 5: Abstract layer-based architecture with specific and common QoO mechanisms for Sensor Webs

5.2 Layer-based observation model

The requirement **R2-levels** advocates for the use of a layer-based observation model. Indeed, each observation level can be seen as the result of the transformation of observations coming from a lower one. For instance, one could say that Raw Data is the result of collection and digitisation of sensor outputs; that Information is obtained by characterising Raw Data with Context, and that Knowledge is achieved through the semantic annotation of Information and the disposal of a base ontology model. System (1) shows a formalised set of equations that describes the transformations needed to obtain the different observation levels.

$$\begin{aligned} f_{digit}(Sensor\ outputs) &= Raw\ Data \\ f_{charac}(Raw\ Data, Context) &= Information \\ f_{sem}(Information, OntoModel) &= Knowledge \end{aligned} \tag{1}$$

Figure 5 shows the example of an abstract layer-based architecture where each layer is responsible for transforming observations according to its main role. Thus, the Raw Data layer performs mostly collection and digitisation, the Information layer performs characterisation and the Semantic layer enables semantic annotation. Each layer produces observations either to upper layer or directly to applications. This situation occurs when a Sensor Web solution does not provide any Information and/or Semantic layers (see the column (4) of Table 2 for more details).

As previously seen in Section 4, a Sensor Web can be partially characterised by some layer-specific and common QoO mechanisms that it provides. Figure 5 shows within which layer these mechanisms are commonly located and used. Regarding layer-specific mechanisms, the Raw Data layer takes normally

care of QoS guarantees. Then, the Information layer may perform Context annotation and compute QoI attributes. Finally, the Semantic layer may annotate observations with semantics and maintain a semantic registry of available sensors (sensor description). Due to their genericity, other common QoO mechanisms (such as Caching, Fusion, Formatting, etc.), can be applied at any layer, on any observation level. Therefore, they are represented as cross-layering mechanisms on Figure 5.

5.3 Mediation

Sensor Webs act as mediators between sensors and applications, orchestrating and coordinating the different exchanges between observation producers and observation consumers. This feature often translates into a negotiation of Service Level Agreements (SLAs), according to consumer needs and producer capabilities. We are convinced that any consumer should be able to express the requested QoO while any observation producer (here the sensors) should be able to express the maximum QoO that it can achieve. In that way, by integrating the QoO notion within SLAs, a Sensor Web may be able to satisfy more easily and more accurately incoming requests given the available sensors.

The requirement **R3-needs** involves that a Sensor Web has to "understand" the expression of consumer needs (with QoO constraints) as well as sensor capabilities. As previously mentioned, many dimensions may be envisioned to express QoO constraints. Our survey shows that mediation within Sensor Webs is almost exclusively achieved with the definition of custom QoO metrics or attributes. To give some example, MASTAQ [29] proposes "standard deviation" and "confidence level" as QoI metrics. MiddleWhere [28] defines "resolution", "confidence" and "freshness" metrics to assess the quality of location information. Wieland et al. [37] use "actuality" and "accuracy" to express QoC. Although it allows finer metric tuning, the use of custom QoO attributes may decrease system interoperability. Indeed, a metric name (confidence level for instance) is rarely sufficient to understand how it is used or how it should be computed. To cope with that issue, Sensor Webs that use custom QoO attributes must provide a good documentation to detail and clarify the metrics used.

In some cases, many observation producers can have similar capabilities. Although they may offer the same kind of observation, they may not operate within the same environment (computing and physical). In that case, the Sensor Web is considered as the entity with the most complete vision of the entire system (oracle) and may perform adaptation or reconfiguration. Depending on its system maturity level, it may also take into account other business rules defined by administrators (such as resource saving or availability).

5.4 Adaptation and reconfiguration

This survey considers adaptation and reconfiguration as two distinct features that Sensor Web should provide (see requirement **R4-adapt**). Figure 6 shows the difference between these two features with a simple example.

Let us consider a use case where an application wants to subscribe to all temperature measurements of a given geographic area. This request leads the Sensor Web to create a specific observation pipeline (Figure 6-a). While it is already receiving some temperature records, the application may decide to modify its needs by asking only for records that indicate a significant heat (to produce heat wave alerts for instance). In this case, we consider that the Sensor Web provides **adaptation** by tuning an existing QoO mechanism (here Filtering with different thresholds) in order to keep meeting consumer needs (Figure 6-b). On the contrary, in case of a sensor failure (Figure 6-c), the Sensor Web is forced to provide **reconfiguration** by selecting other temperature sources (e.g., people who post on Twitter some keywords) or modifying the observation pipeline (e.g., by adding an inference mechanism). While adaptation may be considered as a local strategy (QoO mechanism, agent, etc.), reconfiguration may be more global, involving the cooperation of several entities to keep meeting consumer needs.

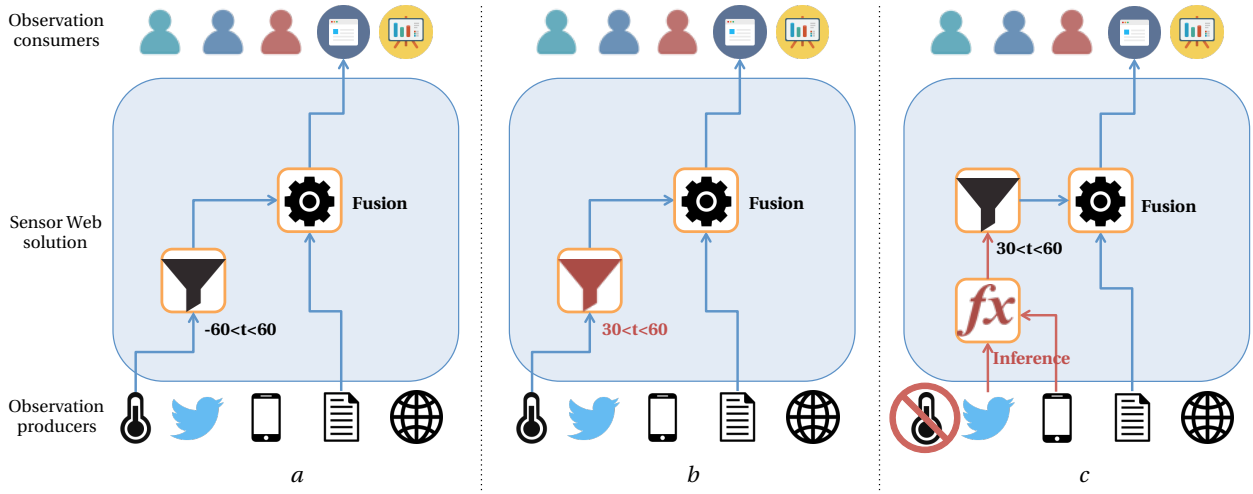

Figure 6: Sensor Web solutions may modify their behaviour to keep meeting consumer needs. a) Nominal state; b) Adaptation (triggered by a change in consumer needs); c) Reconfiguration (triggered by a sensor failure)

Adaptation is, generally, characteristic of decentralised Sensor Web systems, composed of agents or actors with local visions. In SEAMONSTER [36], autonomic adaptation is performed by local agents to meet "mission objectives". In IrisNet [25], another agent-based Sensor Web solution, developers can add logic to sensing agents, writing pieces of code ("senselets") to process Raw Data and provide QoO adaptation. The same mechanism can also be found in the solution of Jiang et al. [26] with the writing of "fuselets". Reconfiguration is mainly achieved with dynamic sensor selection based on consumer needs (like in DQS Cloud [47]) and according to some QoO constraints (like in CASSARAM [48]). However, it can also be implemented by dynamically performing service composition (like in Kibria et al. [51]) or enabling some common QoO mechanisms on the fly (like in the solution of Bouillet et al. [34]).

Finally, it should be noted that a large majority of Sensor Web systems such as OGC SWE [1] or SIXTH [49] do not provide any adaptation or reconfiguration. In this case, these features are delegated to upper applications and it is the responsibility of developers to implement them. In this case, applications are generally strongly coupled to a single system and should implement as many adaptation strategies as there are different Sensor Webs that they use. This way of doing may raise important interoperability and integration issues and may lead to the creation of vertical observation silos.

6 Related Work

QoS within Wireless Sensor Networks has been largely investigated [67, 68]. In these studies, authors present network QoS as a way to improve general information quality. By contrast, this assumption is no longer valid for Sensor Webs. Indeed, within these observation-centric systems, network QoS is rarely sufficient to ensure observation quality. Therefore, other quality dimensions such as QoO need to be considered. The OGC Sensor Web Enablement (SWE), through its members and partners, has published several papers to describe Sensor Web vision. Among these publications, [1] mentions the need to assess QoO within Sensor Web systems.

Up to now, QoO has been mostly investigated within Context-aware systems [18, 69]. Therefore,

Context and QoC have received much attention, especially in the areas of sensor middlewares [70] and IoT [9]. However, Sensor Webs have brought new research challenges like observation heterogeneity and the fact that observation consumers are now applications with specific QoO requirements [71].

The studies presented thus far generally mention QoO as a research challenge while highlighting the importance of considering it within Sensor Webs. As a result, far too little attention has been paid to QoO mechanisms and to their implementation. This trend can be explained by a willingness to let applications manage QoO themselves. Thus, these studies mainly focus on architectural considerations, the benefits of using Sensor Webs or concrete use cases. In [71], Al Nuami et al. compared different Sensor Web architectures and applications. In another survey [72], Maged et al. explain how Sensor Webs can be applied to Crowdsourcing and citizen sensing use cases. Other sensor-related surveys have investigated information quality fields, focusing on QoS [73], QoI [21] or Context information [16]. However, these studies remain very general and only consider the use of quality metrics within "information systems" or "sensor networks".

This survey envisions QoO within Sensor Webs from the perspective of an abstract layer-based architecture with several observation levels and QoO mechanisms. In order to not reinvent the wheel, this survey reuses the "Data, Information, Knowledge, and Wisdom ladder" (DIKW) [11] to refer to the different observation levels. Yet, compared to the DIKW ladder proposed by Seth, this survey only distinguishes three observation levels, considering "Knowledge" and "Wisdom" as a single level. The idea to envision several observation levels is also mentioned in other surveys like in [74] where the National Institute of Standards and Technology (NIST) describes three "perception levels" for sensors ("raw data", "primitive" and "object"). Regarding mechanisms, this survey considers layer-specific and common QoO mechanisms.

To the best of our knowledge, we are the first to provide an extended survey of 29 concrete Sensor Web solutions from a QoO perspective. This survey goes further than the original paper of OGC SWE [1], considering an updated definition for Sensor Webs that also encompasses Cloud-based and IoT-related platforms. Moreover, this survey is a first attempt to describe a set of Sensor Webs by analysing their *Structure*, *Behaviour* and other *System*-related features with regard to QoO. Finally, it also gives directions for the design, implementation and deployment of future Sensor Web solutions that may be useful for researchers and developers.

7 Conclusions and Perspectives

Initially defined by NASA, the Sensor Web vision has evolved over time. This survey envisions a modern and updated definition where a Sensor Web may refer to any Web-based system that bridges the gap between any type of sensors (physical, virtual or logical) and higher-level applications. As observation-centric systems, Sensor Webs must provide and support Quality of Observation (QoO) requirements that may be expressed using several quality dimensions like Data Quality (DQ), Context, Quality of Context (QoC), Quality of Information (QoI) or even Quality of Knowledge (QoK).

After having introduced the Sensor Web vision and given definitions for several quality dimensions that may be used to express and assess QoO, we analyse and evaluate 29 Sensor Web solutions developed between 2003 and 2016. We perform this analysis from a QoO perspective, with a special interest in the implementation of any QoO-related mechanisms. In order to compare the surveyed solutions to each other and exhibit some current trends, we propose a custom taxonomy organised around some *Structural*, *Behavioural* and *System*-related features. We then apply this taxonomy to all surveyed solutions.

To highlight the lessons learned, this survey proposes four requirements that we consider essential for building future adaptive and QoO-aware Sensor Web solutions. These requirements relate to standards, the need for a layer-based architecture with many observation levels, mediation, adaptation and reconfiguration. Among the surveyed solutions, only few meet all requirements previously cited. This leads us to believe that there is still a need for a standardised adaptive and QoO-aware Sensor Web solution.

Besides, a framework to choose and compute meaningful QoO attributes is still missing today.

We advise any developer who wants to design a new Sensor Web to reuse existing standards. However, with only three solutions that comply with OGC SWE standards, our survey confirms that the current trend consists in designing custom and non-standardised lightweight architectures that does not require any learning phase. Ontologies have also shown to be a powerful solution to cope with interoperability issues while allowing inference and higher-level reasoning. In this way, this study has identified the W3C SSN ontology as the most popular ontology for sensors and observations.

Finally, we draw the attention of the reader to the fact that Sensor Webs are critical systems that process and deliver various observations. When applied to sensitive use cases (like health or personal monitoring for instance), additional issues (such as privacy, trust, confidentiality, etc.) need to be considered. In particular, it would be relevant to assess the cost in terms of QoO when dealing with these new issues.

As future work, we are currently developing a platform for QoO assessment *as a Service* that will be applied to a Smart City use case. We hope that this contribution will allow different Smart City stakeholders to assess, better understand and improve QoO in a collaborative way.

8 Acknowledgements

This research was supported in part by the French Ministry of Defence through financial support of the Direction Générale de l'Armement (DGA). The authors would like to thank the reviewers for their valuable feedback and suggestions that have contributed to improve the quality of this paper.

References

[1] A. Bröring, J. Echterhoff, S. Jirka, et al. New Generation Sensor Web Enablement. *Sensors*, 11, (3), pp. 2652–2699, 2011.

[2] A. Sheth, C. Henson, and S. S. Sahoo. Semantic Sensor Web. *IEEE Internet computing*, 12, (4), pp. 78–83, 2008.

[3] A. Bröring, P. Maué, K. Janowicz, D. Nüst, and C. Malewski. Semantically-Enabled Sensor Plug & Play for the Sensor Web. *Sensors*, 11, (8), pp. 7568–7605, 2011.

[4] International Organization for Standardization. Data quality – Part 140: Master data: Exchange of characteristic data: Completeness, . `https://www.iso.org/standard/62395.html`. Accessed May 2017.

[5] Open Geospatial Consortium (OGC). SWE Common Data Model Encoding Standard. `http://www.opengeospatial.org/standards/swecommon`. Accessed May 2017.

[6] International Organization for Standardization. Geographic information – Data quality, . `https://www.iso.org/standard/32575.html`. Accessed May 2017.

[7] K. A. Delin, S. P. Jackson, and R. R. Some. *Sensor Webs*, volume 23 of *NASA Tech Brief*. `http://www.techbriefs.com/component/content/article/1264-ntb/tech-briefs/electronics-and-computers/2227-npo20616?limitstart=0`. Accessed March 2017.

[8] P. M. Teillet. Sensor Webs: A Geostrategic Technology for Integrated Earth Sensing. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 3, (4), pp. 473–480, 2010.

[9] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos. Context Aware Computing for The Internet of Things: A Survey. *IEEE Communications Surveys & Tutorials*, 16, (1), pp. 414–454, 2014.

[10] M. Botts, G. Percivall, C. Reed, and J. Davidson. OGC® Sensor Web Enablement: Overview and high level architecture. In *International conference on GeoSensor Networks*, pp. 175–190, Boston, MA, USA, October 2006. Springer.

[11] A. Sheth. Internet of Things to Smart IoT Through Semantic, Cognitive, and Perceptual Computing. *IEEE Intelligent Systems*, 31, (2), pp. 108–112, 2016.

[12] ITU-T. E.800: Definitions of terms related to Quality of Service. *International Telecommunication Union-Telecommunication Standardisation Sector (ITU-T)*, 2008.

[13] ITU-T. X.641: Information technology - Quality of Service: Framework. *International Telecommunication Union-Telecommunication Standardisation Sector (ITU-T)*, 1997.

[14] Y. Wand and R. Y. Wang. Anchoring Data Quality Dimensions in Ontological Foundations. *Communications of the ACM*, 39, (11), pp. 86–95, 1996.

[15] R. Y. Wang and D. M. Strong. Beyond Accuracy: What Data Quality Means to Data Consumers. *Journal of management information systems*, 12, (4), pp. 5–33, 1996.

[16] A. K. Dey. Understanding and Using Context. *Personal and ubiquitous computing*, 5, (1), pp. 4–7, 2001.

[17] L. Sanchez, J. Lanza, R. Olsen, M. Bauer, and M. Girod-Genet. A Generic Context Management Framework for Personal Networking Environments. In *Mobile and Ubiquitous Systems-Workshops, 2006. 3rd Annual International Conference on*, pp. 1–8, San Jose, CA, USA, July 2006. IEEE.

[18] T. Buchholz and M. Schiffers. Quality of Context: What It Is And Why We Need It. In *Proceedings of the 10th Workshop of the OpenView University Association: OVUA'03*, 2003.

[19] C. Bisdikian, J. Branch, K. K. Leung, and R. I. Young. A letter soup for the quality of information in sensor networks. In *Pervasive Computing and Communications, 2009. PerCom 2009. IEEE International Conference on*, pp. 1–6, Galveston, TX, USA, March 2009. IEEE.

[20] N. Suri, G. Benincasa, R. Lenzi, M. Tortonesi, C. Stefanelli, and L. Sadler. Exploring value-of-information-based approaches to support effective communications in tactical networks. *IEEE Communications Magazine*, 53, (10), pp. 39–45, 2015.

[21] C. Bisdikian, L. M. Kaplan, and M. B. Srivastava. On the Quality and Value of Information in Sensor Networks. *ACM Transactions on Sensor Networks (TOSN)*, 9, (4), pp. 48, 2013.

[22] P. Barnaghi, M. Bermudez-Edo, and R. Tönjes. Challenges for Quality of Data in Smart Cities. *Journal of Data and Information Quality (JDIQ)*, 6, (2-3), pp. 6, 2015.

[23] M. Compton, C. Henson, L. Lefort, H. Neuhaus, and A. Sheth. A Survey of the Semantic Specification of Sensors. In *Proceedings of the 2nd International Conference on Semantic Sensor Networks*, volume 522, pp. 17–32, Washington DC, DC, USA, October 2009. CEUR-WS.org.

[24] L. Rao and K.-M. Osei-Bryson. Towards defining dimensions of knowledge systems quality. *Expert Systems with Applications*, 33, (2), pp. 368–378, 2007.

[25] P. B. Gibbons, B. Karp, Y. Ke, S. Nath, and S. Seshan. IrisNet: an architecture for a worldwide sensor Web. *IEEE Pervasive Computing*, 2, (4), pp. 22–33, 2003.

[26] G. Jiang, W. W. Chung, and G. Cybenko. Semantic agent technologies for tactical sensor networks. In *SPIE's AeroSense 2003 (OR03)*, pp. 311–320, Orlando, FL, USA, April 2003. International Society for Optics and Photonics.

[27] A. Ranganathan and R. H. Campbell. A Middleware for Context-Aware Agents in Ubiquitous Computing Environments. In *ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing*, pp. 143–161, Rio de Janeiro, Brazil, June 2003. Springer.

[28] A. Ranganathan, J. Al-Muhtadi, S. Chetan, R. Campbell, and M. D. Mickunas. MiddleWhere: A Middleware for Location Awareness in Ubiquitous Computing Applications. In *Proceedings of the 5th ACM/IFIP/USENIX International Conference on Middleware*, Middleware '04, pp. 397–416, Toronto, Canada, October 2004. Springer.

[29] I. Hwang, Q. Han, and A. Misra. MASTAQ: A middleware architecture for sensor applications with statistical quality constraints. In *3rd IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom 2005)*, pp. 390–395, Kauai Island, Hawaii, March 2005. IEEE.

[30] K. Aberer, M. Hauswirth, and A. Salehi. Middleware support for the Internet of Things. In *Proceedings of 5. GI/ITG KuVS Fachgespraech-Drahtlose Sensornetze*, pp. 15–19, Berlin, Germany, September 2006.

[31] C. Jacob, D. Linner, S. Steglich, and I. Radusch. Bio-inspired Context Gathering in Loosely Coupled Computing Environments. In *Bio-Inspired Models of Network, Information and Computing Systems, 2006. 1st*, pp. 1–6, York, UK, 2006. IEEE.

[32] D. Moodley and I. Simonis. A New Architecture for the Sensor Web: The SWAP Framework. In *Proceedings of 5th International Semantic Web Conference (ISWC 2006)*, volume LNCS 4273, Athens, GA, USA, 2006.

[33] W. I. Grosky, A. Kansal, S. Nath, J. Liu, and F. Zhao. SenseWeb: An Infrastructure for Shared Sensing. *IEEE multimedia*, 14, (4), 2007.

[34] E. Bouillet, M. Feblowitz, Z. Liu, A. Ranganathan, A. Riabov, and F. Ye. A Semantics-Based Middleware for Utilizing Heterogeneous Sensor Networks. In *International Conference on Distributed Computing in Sensor Systems (DCOSS'07)*, pp. 174–188, Santa Fe, New Mexico, USA, June 2007. Springer.

[35] P. Hu, J. Indulska, and R. Robinson. An Autonomic Context Management System for Pervasive Computing. In *Pervasive Computing and Communications, 2008. PerCom 2008. Sixth Annual IEEE International Conference on*, pp. 213–223, Hong Kong SAR, China, March 2008. IEEE.

[36] J. S. Kinnebrew, W. R. Otte, N. Shankaran, G. Biswas, and D. C. Schmidt. Intelligent Resource Management and Dynamic Adaptation in a Distributed Real-time and Embedded Sensor Web System. In *Object/Component/Service-Oriented Real-Time Distributed Computing, 2009. ISORC'09. IEEE International Symposium on*, pp. 135–142, Tokyo, Japan, March 2009. IEEE.

[37] M. Wieland, U.-P. Käppeler, P. Levi, F. Leymann, and D. Nicklas. Towards Integration of Uncertain Sensor Data into Context-aware Workflows. In *GI Jahrestagung*, pp. 2029–2040. Citeseer, 2009.

[38] M. Pathan, K. Taylor, and M. Compton. Semantics-based plug-and-play configuration of sensor network services. In *SSN'10 Proceedings of the 3rd International Conference on Semantic Sensor Networks*, volume 668, pp. 17–32, Shanghai, China, October 2010. CEUR-WS.org.

[39] D. Romero, R. Rouvoy, L. Seinturier, S. Chabridon, D. Conan, and N. Pessemier. Enabling Context-Aware Web Services: A Middleware Approach for Ubiquitous Environments. In M. Sheng, J. Yu, and S. Dustdar, editors, *Enabling Context-Aware Web Services: Methods, Architectures, and Technologies*, pp. 113–135. Chapman and Hall/CRC, 2010.

[40] T. Teixeira, S. Hachem, V. Issarny, and N. Georgantas. Service Oriented Middleware for the Internet of Things: A Perspective. In *Towards a Service-Based Internet: 4th European Conference, ServiceWave 2011. Proceedings*, pp. 220–229, Poznan, Poland, October 2011. Springer.

[41] C. J. Matheus, A. Boran, D. Carr, et al. Semantic Network Monitoring and Control over Heterogeneous Network Models and Protocols. In *International Conference on Active Media Technology*, pp. 433–444, Macau, China, December 2012. Springer.

[42] D. Le-Phuoc, H. Q. Nguyen-Mau, J. X. Parreira, and M. Hauswirth. A middleware framework for scalable management of linked streams. *Web Semantics: Science, Services and Agents on the World Wide Web*, 16, pp. 42–51, 2012.

[43] C. Bizer, T. Heath, and T. Berners-Lee. Linked Data-the story so far. *Semantic Services, Interoperability and Web Applications: Emerging Concepts*, pp. 205–227, 2009.

[44] K. Da, P. Roose, M. Dalmau, J. Nevado, and R. Karchoud. Kali2Much: a context middleware for autonomic adaptation-driven platform. In *Proceedings of the 1st ACM Workshop on Middleware for Context-Aware Applications in the IoT*, pp. 25–30, Bordeaux, France, December 2014. ACM.

[45] S. Hachem, A. Pathak, and V. Issarny. Service-Oriented Middleware for the Mobile Internet of Things: A Scalable Solution. In *IEEE GLOBECOM: Global Communications Conference*, Austin, TX, USA, December 2014.

[46] P. Marie, L. Lim, A. Manzoor, S. Chabridon, D. Conan, and T. Desprats. 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'14)*, pp. 13–18, Bordeaux, France, December 2014. ACM.

[47] A. Kothari, V. Boddula, L. Ramaswamy, and N. Abolhassani. DQS-Cloud: A Data Quality-Aware autonomic cloud for sensor services. In *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2014 International Conference on*, pp. 295–303. IEEE, October 2014.

[48] C. Perera, A. Zaslavsky, C. H. Liu, M. Compton, P. Christen, and D. Georgakopoulos. Sensor Search Techniques for Sensing as a Service Architecture for the Internet of Things. *IEEE Sensors Journal*, 14, (2), pp. 406–420, 2014.

[49] D. Carr. *The SIXTH Middleware: sensible sensing for the sensor web*. PhD thesis, University College Dublin, 2015.

[50] J. Soldatos, N. Kefalakis, M. Hauswirth, et al. OpenIoT: Open Source Internet-of-Things in the Cloud. In *Interoperability and Open-Source Solutions for the Internet of Things: International Workshop, FP7 OpenIoT Project, Held in Conjunction with SoftCOM 2014, Invited Papers*, volume 9001, pp. 13–25, Split, Croatia, September 2015. Springer.

[51] M. G. Kibria, S. M. M. Fattah, K. Jeong, I. Chong, and Y.-K. Jeong. A User-Centric Knowledge Creation Model in a Web of Object-Enabled Internet of Things Environment. *Sensors*, 15, (9), pp. 24054–24086, 2015.

[52] D. Puiu, P. Barnaghi, R. Tönjes, et al. CityPulse: Large Scale Data Analytics Framework for Smart Cities. *IEEE Access*, 4, pp. 1086–1108, 2016.

[53] S. Ramalingam and L. Mohandas. A Fuzzy Based Sensor Web for Adaptive Prediction Framework to Enhance the Availability of Web Service. *International Journal of Distributed Sensor Networks*, 12, (2), 2016.

[54] E. Brewer. CAP twelve years later: How the" rules" have changed. *Computer*, 45, (2), pp. 23–29, 2012.

[55] S. H. Javadi and A. Peiravi. Fusion of weighted decisions in wireless sensor networks. *IET Wireless Sensor Systems*, 5, (2), pp. 97–105, 2015.

[56] B. Jacob, R. Lanyon-Hogg, D. K. Nadgir, and A. F. Yassin. A Practical Guide to the IBM Autonomic Computing Toolkit. *IBM Redbooks*, 4, pp. 10, 2004.

[57] J. O. Kephart and D. M. Chess. The vision of Autonomic Computing. *Computer*, 36, (1), pp. 41–50, 2003.

[58] C. C. Aggarwal, N. Ashish, and A. Sheth. The Internet of Things: A Survey from the Data-Centric Perspective. In *Managing and mining sensor data*, pp. 383–428. Springer, 2013.

[59] L. Atzori, A. Iera, and G. Morabito. The Internet of Things: A survey. *Computer Networks*, 54, (15), pp. 2787–2805, October 2010.

[60] A. P. Abidoye and I. C. Obagbuwa. Models for integrating wireless sensor networks into the Internet of Things. *IET Wireless Sensor Systems*, 2017.

[61] EU FP7. IOT-A: Internet of Things Architecture. `http://www.iot-a.eu/public`. Accessed March 2017.

[62] A. Bröring, K. Janowicz, C. Stasch, and W. Kuhn. Semantic Challenges for Sensor Plug and Play. In *International Symposium on Web and Wireless Geographical Information Systems*, pp. 72–86. Springer, 2009.

[63] C. A. Henson, J. K. Pschorr, A. P. Sheth, and K. Thirunarayan. SemSOS: Semantic sensor Observation Service. In *Collaborative Technologies and Systems, 2009. CTS'09. International Symposium on*, pp. 44–53. IEEE, 2009.

[64] W3C SSN Incubator Group. Review of Sensor and Observations Ontologies, 2011. `https://www.w3.org/2005/Incubator/ssn/wiki/Review_of_Sensor_and_Observations_Ontologies`. Accessed March 2017.

[65] M. Compton, P. Barnaghi, L. Bermudez, et al. The SSN ontology of the W3C semantic sensor network incubator group. *Web semantics: science, services and agents on the World Wide Web*, 17, pp. 25–32, 2012.

[66] R. G. Raskin and M. J. Pan. Knowledge representation in the semantic web for Earth and environmental terminology (SWEET). *Computers & geosciences*, 31, (9), pp. 1119–1125, 2005.

[67] D. Chen and P. K. Varshney. QoS Support in Wireless Sensor Networks: A Survey. In *International conference on wireless networks*, volume 233, pp. 1–7, 2004.

[68] I. F. Akyildiz, T. Melodia, and K. R. Chowdhury. A survey on wireless multimedia sensor networks. *Computer networks*, 51, (4), pp. 921–960, 2007.

[69] C. Bettini, O. Brdiczka, K. Henricksen, et al. A survey of context modelling and reasoning techniques. *Pervasive and Mobile Computing*, 6, (2), pp. 161–180, 2010.

[70] K. Sheikh, M. Wegdam, and M. Van Sinderen. Middleware Support for Quality of Context in Pervasive Context-Aware Systems. In *Pervasive Computing and Communications Workshops, 2007. PerCom Workshops' 07. Fifth Annual IEEE International Conference on*, pp. 461–466, White Plains, NY, USA, March 2007. IEEE.

[71] K. Al Nuaimi, M. Al Nuaimi, N. Mohamed, I. Jawhar, and K. Shuaib. Web-based Wireless Sensor Networks: A Survey of Architectures and Applications. In *Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication*, p. 113, Kuala Lumpur, Malaysia, February 2012. ACM.

[72] M. N. K. Boulos, B. Resch, D. N. Crowley, et al. Crowdsourcing, citizen sensing and sensor web technologies for public and environmental health surveillance and crisis management: trends, OGC standards and application examples. *International journal of health geographics*, 10, (1), pp. 1, 2011.

[73] P. Bellavista, A. Corradi, and A. Reale. Scalable Stream Processing with Quality of Service for Smart City Crowdsensing Applications. *EAI Endorsed Transactions on Mobile Communications and Applications*, 13, (3), 2013.

[74] R. Eastman, C. Schlenoff, S. Balakirsky, and T. Hong. A Sensor Ontology Literature Review. Technical Report NIST IR 7908, National Institute of Standards and Technology, 2013. `http://nvlpubs.nist.gov/nistpubs/ir/2013/NIST.IR.7908.pdf`. Accessed March 2017.

Table 2: Comparison of 29 Sensor Web solutions designed between 2003 and 2016. Abbreviations listed in Table 3.

			System		Structure		Behaviour	
Sensor Web solution	**Reference**	**Year**	**Obs. levels supported**	**System maturity level**	**Architecture styles**	**Standard-compliant**	**Layer-specific mech.**	**Common QoO mech.**
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
IrisNet	[25]	2003	RD, I	2	Dist, Comp	×	-	Cach, Filt
Jiang et al.	[26]	2003	RD, I	3	Dist, Comp	∽	Sens	Fu, Filt
Ranganathan et al.	[27]	2003	I	3	Dist, Comp	∽	Obs	ML
MiddleWhere	[28]	2004	I	3	Data, Dist	×	QoI	Fu, Filt
MASTAQ	[29]	2005	I	4	Data, Hiera	×	QoI	ML
GSN	[30]	2006	RD, I	3	Flow, Dist, Comp	∽	Cont	Fu, Form, Filt
BIONETS	[31]	2006	I	2	Data, Dist	∽	QoI, Obs	Cach
SWAP	[32]	2006	K	2	Hiera, Dist, Comp	✓	Sens, Obs	Fu, Form
SenseWeb	[33]	2007	RD, I	3	Hiera	×	QoS, QoI	Cach, Form
Bouillet et al.	[34]	2007	K	4	Flow, Dist	∽	Cont, QoI, Sens, Obs	Form, Filt
AcoMS	[35]	2008	I	5	Flow, Data	∽	QoS, QoI, Sens	Form, Filt
SEAMONSTER	[36]	2009	RD	5	Dist, Comp	×	QoS, Cont	-
Wieland et al.	[37]	2009	I	4	Flow, Data, Hiera	∽	Cont, QoI	Fu, Filt
Pathan et al.	[38]	2010	K	5	Dist	∽	Cont, Sens, Obs	-
CAPPUCINO	[39]	2010	-	5	Data, Comp	∽	-	Fu, Filt
OGC SWE	[1]	2011	RD, I	2	Dist	✓	-	Fu, Filt
Teixera et al.	[40]	2011	I	3	Flow, Dist	∽	Obs, Sens	Fu, Filt, ML
SNoMAC	[41]	2012	K	1	Hiera	∽	QoS, Sens	-
LSM	[42]	2012	K	1	Data, Dist	∽	Obs, Sens	Cach, Fu, Filt
Kali2Much	[44]	2014	I	5	Flow, Data	×	Obs, Sens	Form, Filt
MobIoT	[45]	2014	I, K	3	Data, Dist	∽	Obs, Sens	Fu, Form, ML
INCOME	[46]	2014	I	4	Flow, Data	×	Cach	Fu, Form, Filt
DQS Cloud	[47]	2014	RD, I	4	Data	×	QoS, Cont, QoI	Filt
CASSARAM	[48]	2014	K	4	Data	∽	Cont, QoI, Obs, Sens	-
SIXTH	[49]	2015	I	4	Dist, Comp	∽	Cont	Form, Filt
OpenIoT	[50]	2015	K	4	Data, Dist	∽	QoS, Cont, QoI, Obs, Sens	Fu, Filt
Kibria et al.	[51]	2015	K	4	Data, Hiera	×	QoI, Obs, Sens	Cach, ML
CityPulse	[52]	2016	K	5	Data, Dist	∽	QoS, Cont, QoI, Obs, Sens	Cach, Fu, Form, Filt, ML
FAPFEA	[53]	2016	-	5	Dist	✓	-	ML

Table 3: Abbreviations used for the surveyed solutions

#	**Column name**	**Features**
(4)	Observation levels supported	Raw Data (RD) Information (I) Knowledge (K)
(5)	System maturity level	Basic (1) Managed (2) Predictive (3) Adaptive (4) Autonomic (5)
(6)	Architectural styles and paradigms	Data flow (Flow) Data-centred (Data) Hierarchical (Hiera) Distributed (Dist) Component-based (Comp)
(7)	Standard-compliant	Standardised solution (✓) Partially standardised solution (⌣) Non-standardised solution (×)
(8)	Layer-specific mechanisms	Network QoS guarantees (QoS) Context annotation (Cont) QoI computation (QoI) Semantic-based sensor description (Sens) Semantic-based observation annotation (Obs)
(9)	Common QoO mechanisms	Caching (Cach) Fusion (Fu) Formatting (Form) Filtering (Filt) Machine Learning (ML)