Knowledge-Based Systems 109 (2016) 122–136

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

A knowledge-based resource discovery for Internet of Things☆

Charith Perera[a,*], Athanasios V. Vasilakos[b]

[a] Centre for Research in Computing, The Open University, Milton Keynes, UK
[b] Dept of Computer Science, Electrical and Space Engineering, Lulea University of Technology, Lulea, Sweden

ARTICLE INFO

Article history:
Received 22 January 2016
Revised 23 June 2016
Accepted 26 June 2016
Available online 29 June 2016

Keywords:
Internet of Things
Middleware
Semantic knowledge
IoT resource composition

ABSTRACT

In the sensing as a service paradigm, Internet of Things (IoT) Middleware platforms allow data consumers to retrieve the data they want without knowing the underlying technical details of IoT resources (i.e. sensors and data processing components). However, configuring an IoT middleware platform and retrieving data is a significant challenge for data consumers as it requires both technical knowledge and domain expertise. In this paper, we propose a knowledge driven approach called Context Aware Sensor Configuration Model (CASCOM) to simplify the process of configuring IoT middleware platforms, so the data consumers, specifically non-technical personnel, can easily retrieve the data they required. In this paper, we demonstrate how IoT resources can be described using semantics in such away that they can later be used to compose service work-flows. Such automated semantic-knowledge-based IoT resource composition approach advances the current research. We demonstrate the feasibility and the usability of our approach through a prototype implementation based on an IoT middleware called Global Sensor Networks (GSN), though our model can be generalized to any other middleware platform.

1. Introduction

The Internet of Things (IoT) [2] envisions connecting billions of smart devices to the Internet. It provides a networked infrastructure that enables things to be connected anytime, anyplace, with anything and anyone, ideally using any path, any network and any service [31]. These smart devices should be smoothly integrated within Future Internet (FI) service delivery models such as sensing as a service. The *things*[1] in IoT are accompanied with sensors and actuators. It is estimated that there are about 1.5 billion Internet-enabled PCs and over 1 billion Internet-enabled mobile phones today. By 2020, there will be 50 to 100 billion devices connected to the Internet [31]. Since these smart devices comprise sensors, it is evident that there would be many sensors deployed around us in the future. Even today, sensors are used in many domains such as agriculture, environmental monitoring, and manufacturing [25].

In order to analyse and understand a given phenomenon extensively, data generated from appropriate sensors needs to be fed into more sophisticated data analysis applications. These applications are designed to produce certain results once they are given required sensor data as inputs. IoT middleware solutions simplify the retrieval of data from sensors for these applications by acting as a mediator between the hardware layer and the application layer. In order to perform these bindings, middleware solutions need to be configured depending on the context information and user requirements. Our objective is to automate and simplify the configuration of IoT middleware platforms and improve their usability so both IT experts and non-IT experts can use them efficiently and effectively.

There are several characteristics we have identified as important for developing a model for IoT that provisions sensing as a service by formulating and composing multiple types of sensor as well as different filtering, fusing, and reasoning mechanisms together on-demand. The core features of the proposed model are as follows:

- *Autonomic:* The model should support the dynamic composition of internet-connected objects, in response to dynamically defined end-users' requests. To this end, we have incorporated semantic knowledge [30], along with automated reasoning algorithms for orchestrating sensors, and data processing mechanisms [25], according to the data consumer requests.
- *Utility based:* The proposed model should deliver services according to a utility computing model [8,22]. It should offer sensing capability as a service[26] over dynamically created and

☆ An earlier version of this work has been published in the Proceedings of the 9th International Conference on Semantics, Knowledge & Grids (SKG).

* Corresponding author.
E-mail addresses: charith.perera@ieee.org, ngcharithperera@gmail.com (C. Perera), vasilako@ath.forthnet.gr (A.V. Vasilakos).

[1] We use terms *objects, things, smart objects, devices, nodes* to give the same meaning as they are frequently used in IoT literature interchangeably.

http://dx.doi.org/10.1016/j.knosys.2016.06.030
0950-7051/

configured solutions[2] that are custom generated for each consumer request. Sensor data consumers (users) should be allowed to make the decisions on the characteristic of the solution (e.g. accuracy, reliability, latency and so on). Orchestration of IoT resources (i.e sensors and data processing components) in the cloud environment at runtime is an important functionality [5]. The dynamism implies the capability of adapting to resources changes in volatile environments where sensors and data processing components may be added or removed from the system over time. This means that new solutions will be able to compose together over time due availability of new resources.

- *Scalability and flexibility:* The proposed solution should be flexible so data processing components and sensors can be added over time [33]. Further, the proposed solution should be scalable so any number of IoT resources can be supported. Such ability increases the types of consumer requests that can be fulfilled. Further, it increases the number of different solutions that can be formulated to accommodate a single request. Possibility of creating multiple different solutions will increase the choice and control consumer have. Finally, the proposed model and its algorithms should be independent from the data (i.e. descriptions of IoT resources) so adding a new resource does not require changes to be made into the system.
- *Ease of use / reduced learning curve:* One of the primary goals of an IoT middleware is to enable users to retrieve data quickly without dealing with complex hardware or software level configurations. It is important to make all the process simplified so a non-technical personal (e.g, biologist) can use these IoT middleware platforms to collect the data they need with minimum effort.

1.1. Motivation

Over the last few years, we have seen more and more IoT middleware platforms making their way in the marketplace. Large number of sensors are expected to connect to these middleware platforms. Further, variety of different IoT applications are expected to be built on top of these middleware platforms. These IoT applications have different types of algorithms that analyse data built into them. These algorithms are nothing but some type black boxes that take specific type of inputs and generate specific type of outputs. One of the main responsibilities of an IoT middleware is to hide and abstract the connectivity and communication details of sensors and support the users to retrieve the data streams they required to be fed into their application easily and quickly.

Data processing components can also be used to build these required data streams as we later discuss in this paper. Currently, it is difficult to configure IoT middleware platforms in a way that they produces a certain data stream that is required by an IoT application. The challenges are discussed in Section 2. To make IoT middleware configuration easier, we propose a knowledge driven approach called Context Aware Sensor Configuration Model (CASCOM) to simplify the process of configuring IoT middleware platforms, so the data consumers, specially non-technical personnel, can easily retrieve the data they required.

1.2. Main contributions

The contributions of our paper are as follows:

- We propose a IoT configuration model called CASCOM to enrich existing IoT middleware platforms. This model helps non-IT experts to configure sensors and data processing components with less effort.
- CASCOM is completely driven by semantically enriched IoT resource descriptions at the back end. Therefore, new sensors and data processing components can be added at any time. No changes are required in the application from an algorithmic perspective.
- CASCOM provides an easy way to construct the data streams required by the data consumers by selecting questions and answering them.
- CASCOM automatically highlights to users on potential secondary context information that can be derived from existing primary context.
- Finally, CASCOM informs the users regarding potential sensor and data processing components that can be added to the system in order to enhance the ability to serve the user requests.
- CASCOM uses ontologies to model semantics where three ontologies capture the relevant knowledge collectively. The usage of ontologies help to automated composition and reasoning process. More importantly, modelling new knowledge is very easy due to the adoption of ontology based knowledge modelling technique. Specifically, we employ two existing ontologies, namely SCO [9], and SSN ontology [10] and developed our own ontology called QA+TDO. New knowledge can be added to these existing models easily using the proposed tool.

We explain all the above mentioned contributions in detail throughout the paper. The rest of this paper is structured as follows. Section 2 presents the background and related work. First, we briefly introduce an IoT reference architecture and its characteristics. Then, we explain where our proposed model fits in such an architecture. Later, we review some related work and compare them with our own to highlight the similarities and differences. The research challenges are discussed in Section 3. We have used a real world use-case scenario from the agriculture domain to explain the research problem in detail. Our research question is *'How to develop a model that allows data consumers (i.e, non-IT and IT experts) to configure IoT middleware platforms by discovering and composing IoT resources (i.e, sensors and data processing components) effortlessly?'*. Once the configuration is completed, the IoT middleware platform should produce the data streams that the data consumers have requested. Data consumers can use these data streams to achieve their own objectives.

Subsequently, we explain the importance of resource discovery and composition in IoT domain and why it need to be knowledge-driven. Architectural designs are presented in Section 4. We propose our solution, CASCOM, which consists of six phases where each phase is explained in detail with relevant algorithms and examples. Section 6 presents the implementation and experimentation details. We evaluate the proposed model from both computational (i.e. storage requirements, data model loading time, query time) and usability point of views, presenting our findings in Section 7. We explain why our proposed model is feasible and how it helps users to configure IoT middleware platforms easily. Finally, we conclude the paper in Section 8.

2. Background and related work

2.1. Background

In this section, we review a number of related work and discuss the problem domain in detail. Broadly, configuration in IoT paradigm can be categorized into two areas: *sensor-level* configuration and *system-level* configuration. Sensor-level configuration [13] focuses on changing a sensor's behaviour by configuring its embedded software parameters such as sensing schedule, sampling

[2] Solution is a combination of sensors and data processing components that can be composed together in order to satisfy a user requirement.

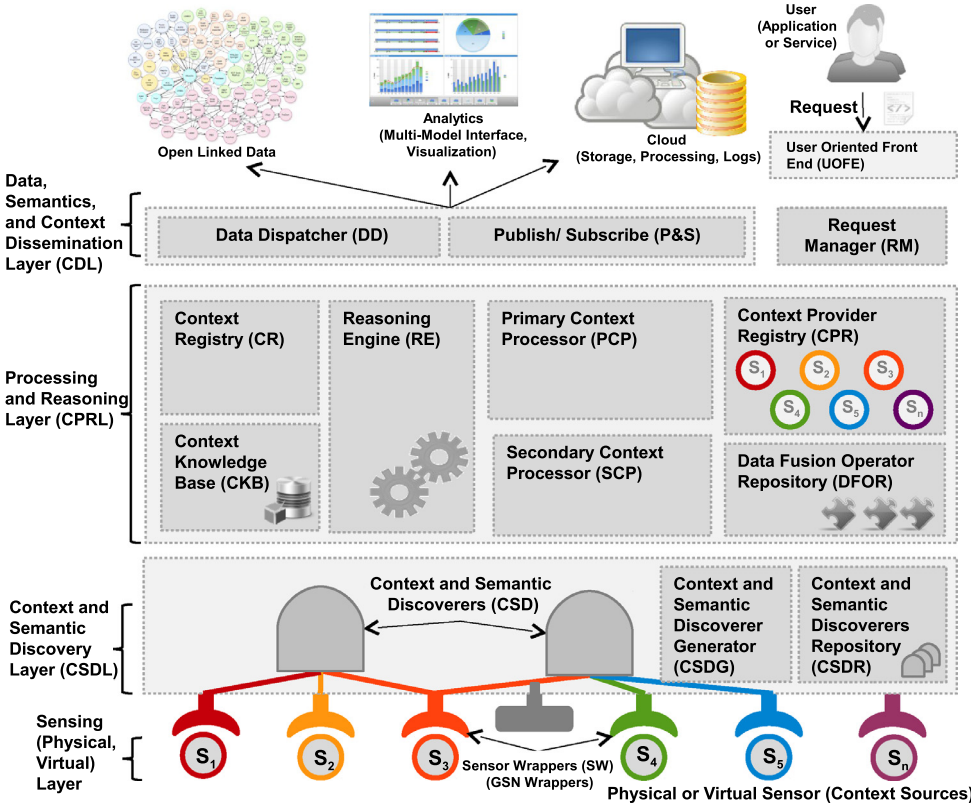

Fig. 1. Internet of Things reference architecture. Our proposed model, CASCOM, fits within the Reasoning Engine (RE) block of the architecture. The details of this architecture is discussed in detail in [24].

rate, data communication frequency, communication patterns and protocols. In this paper, we focus on developing a system-level configuration model for IoT midddleware platforms. System-level configuration focuses on changing the behaviour of IoT middleware systems by configuring internal software components. Specifically, our proposed model identifies, composes, and configures both sensors and data processing components in order to create the data streams based on user requirements.

We start by briefly introducing a reference architecture for IoT middleware. Our reference architecture, a detailed description of which is given in [24], is presented in Fig. 1. The details are presented in [24]. Even though the details of this reference architecture are out of scope of this paper, we would like to briefly introduce some of the major responsibilities of an IoT middleware and its different components. The objective of an IoT middleware from users' perspective is to collect sensor data streams so they can inject them into an application that is capable of performing analysis [4]. A data stream is simply a set of data items that is captured and transferred to the users sequentially and continuously at certain intervals (e.g. every 5 s, every 2 h). A sample data stream is illustrated in Fig. 2. A data stream may consist of one type of data (e.g. as illustrated in data stream 2 in Fig. 2) or multiple types of data (e.g. as illustrated in data stream 1 in Fig. 1).

The reference architecture illustrated in Fig. 1 consists of four layers: Data, Semantics, and Context Dissemination Layer (DSCDL), Context Processing and Reasoning Layer (CPRL), Context and Semantic Discovery Layer (CSDL), and Sensor Data Acquisition Layer (SDAL). Data, Semantics, and Context Dissemination Layer (DSCDL) is responsible for user management. The components belonging to this layer are data dispatcher, request manager, and publish/subscribe. Typically, users will not know about the technical details of the sensors or data processing components. They only know about the problem they need to solve. therefore, users need

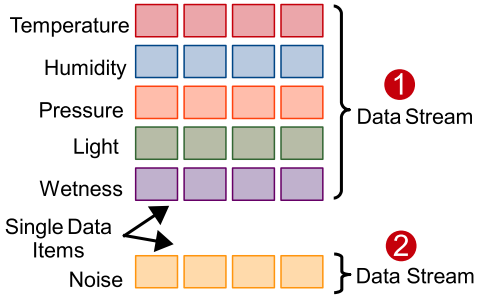

Fig. 2. Data stream.

to be provided with the easy-to-use mechanisms to express their requirements in high-level without requiring technical knowledge.

The Processing and Reasoning Layer (CPRL) is responsible for data processing, reasoning, fusing, knowledge generating and storing. In this layer data processing components are organized into work-flows in such a way that they collectively produce the data streams required by the consumers. Context and Semantic Discovery Layer (CSDL) is responsible for managing context and generating secondary context information from primary context information. Sensor Data Acquisition Layer (SDAL) is responsible for acquiring data. This layer communicates with hardware and software sensors and retrieves sensor data into IoT middleware.

In this reference architecture, data processing components sit within the Data Fusion Operator[3] (DFO) registry. Similarly, Context Provider Registry (CPR) keeps track of the data items captured by

[3] DFO is also called the data processing component.

the sensors. Reasoning Engine (RE) is responsible for building the above mentioned work-flow solutions to satisfy user requests.

The challenge of configuring an IoT middleware solution at run time can be understood by analysing an existing middleware such as Global Sensor Networks (GSN) [1]. Some of the key challenges are as follows.

- Users need to know the low-level details such as data types and measurement units of the sensors in order to request them manually.
- It is extremely difficult to memorise different combinations of sensor data types that can be used to fulfil user requirements (e.g. which sensors need to be composed together to detect an event?). In particular, domain knowledge (e.g., relating to agriculture) is difficult to memorise when there are multiple ways of building a given data stream.
- Users need to know the availability of data processing components, their input/output data types and their capabilities to develop a strategy. Data processing operations need to be applied on data in the correct sequence.
- There is no way to find out the strategies to overcome the issues when existing hardware resources (i.e. existing sensors) and software resources (i.e. data processing components) are incapable of producing the results that users required.
- Further, the solutions designed by users may not be optimal (e.g. due to the variability of hardware and software costs).

An ideal IoT middleware configuration model should address all the above mentioned challenges. The proposed configuration model, CASCOM, is applicable towards several other emerging paradigms, such as sensing as a service [35]. Our proposed solution combines technologies from different research areas such as IoT middleware, semantic technologies, software component composition, and context-aware computing. We discuss major related research efforts in the remainder of this section.

2.2. Related work

Microsoft *SensorMap* [18] (sensormap.org) is a data sharing and visualization framework. It is a peer produced sensor network that consists of sensors deployed by contributors around the world. *SensorMap* mashes up sensor data on a map interface. Then, it allows to selectively query sensors and visualize data. Our approach completely automates the configuration process by eliminating the requirement of hand picking sensors. *Linked Sensor Middleware* (LSM) [29] (lsm.deri.ie) is a platform that provides wrappers for real time data collection and publishing. It also provides a web interface for sensor search, linked stream data query, data annotation and visualisation. *LSM* mainly focuses on linked data publishing. Sensor selection needs to be done manually in order to retrieve sensor data. *Xively* (Xively.com) is a platform for Internet of Things devices. *Xively* allows different data sources to be connected to it. Then, it provides functionalities such as event triggering and data filtering. It acts as a mediator between sensors and applications where users need to manually select and configure sensors. *HyperCat* (hypercat.io) is an open, lightweight JSON-based hypermedia catalogue format for exposing collections of URIs. *HyperCat* has proposed the notion of describing resources in a semantic way. These descriptions are designed for exposing information about IoT assets over the web. *HyperCat* provides a standard mechanism for developers to publish linked-data descriptions of resources.

Context-awareness is a critical functionality that needs to be embedded into IoT middleware solutions [25]. Context information (e.g. accuracy, reliability, cost) plays a significant role in selecting sensors and data processing components [23]. To support this, CASCOM provides context discovery functionalities by using semantic knowledge and fusing raw sensor data. The *SensorMashup* [28] platform offers a visual composer for sensor data streams. Data sources and intermediate analytical tools are described by reference to an ontology, enabling an integrated discovery mechanism for such sources. Selection of data sources and analytical tools based on user requirement need to be done manually by users. Khemakhem et al. [14] use multiple ontologies to discover and compose software components by focusing on non-functional proprieties. In web service composition domain, service composition means composing a larger service by combining many smaller services. This is the same principle used when composing a larger software component from many smaller components.

Web service (WS) composition using ontologies [17] is similar to IoT resource composition performed in CASCOM from a functional point of view but different from an implementation and execution point of view. Web services composition domain only involves in combining multiple software components. In contrast, CASCOM needs to deal with both hardware and software components in its configuration model. We present a comparison of WS composition and IoT resource composition in Table 1.

Leitner et al. [16] have proposed a cost-based service composition model to support service level agreements (SLA) in manufacturing domain. Similar to our approach, in SLAs, customers are allowed to express their requirements and expectations (e.g. monetary costs, time to deliver, quality). Based on the customer needs, different service providers will be used to accommodate the request order (e.g. normal shipping or express shipping). However, these composition are done based on predefined business rules. Haddad et al. [11] have addressed the issue of selecting and composing Web services not only according to their functional requirements but also to their transactional properties and QoS characteristics. Though QoS characteristics are important in IoT resource composition domain, transactional properties are less relevant. The reason is that IoT resource composition does not need to support compensations or undoing transactions. Bronsted et al. [7] has mentioned that *'Only 10 cases use scenario-based evaluation, which is most realistic because it involves actual use by users. So, for many of the mechanisms, there's weak empirical support for the claim that they work in realistic settings'*. This is one of the reasons we evaluate our approach using use-case and also described the processes and techniques using real-world applications. Kritikos and Plexousakis [15] have discussed quality of service and its importance towards web service discovery. They recognize QoS as a set of performance and domain-dependent attributes that has a substantial impact on WS requesters' expectations. This is also similar in IoT resource composition domain as well.

Several projects [9] have designed and developed ontologies to describe software components. Such approaches have helped them to perform dynamic composition of software components. A process of software component matching using ontologies has been explained in [21]. In our work we employed the Software Component Ontology discussed in [9]. Semantic Sensor Ontology (SSNO) [10] also allowed us to model sensor descriptions. Noguchi et al. [19] have proposed a mechanism that generates connection between different software components in order to process sensor data and detect events. In contrast, our objective is to produce the data streams required by the users so they can be further analysed extensively using sophisticated applications.

3. Research challenges

This section describes and analyses the research challenges in detail with concrete examples and scenarios. Fig. 3 illustrates the problem in general. The explanations are based on agriculture and environmental monitoring domains. The proposed solution helps data consumers to overcome the difficulties listed in Section 2. Our research question is *'How to develop a model that allows non-IT*

Table 1
Comparison of web services composition and IoT resource composition Domains. In summary, web services selection is based on virtual capabilities and characteristics where IoT resources selection is based on both physical and virtual capabilities and characteristics. As a result, IoT resources are much complex elements to be selected and composed autonomously than web services.

	Web service domain [36]	IoT domain
Similarities	• Consuming single web service may not create significant value. Therefore, web services selection and composition is critical to generate value. • Many alternative web services are available to use • Can be found through directory services • Quality of services matters [34]. • There are free as well as paid services.	• Collecting data from a single sensor may not create significant value. Therefore, sensor selection and composition is critical to generate value. • Many alternative sensors will be available to use • Middleware solutions such as OpenIoT and GSN will play a mediator roles between sensors and sensor data consumers • Quality of sensors (and data) matters • There will be free as well as paid sensors
Differences	• Web service compose with other web services in to work-flows. • Largely guided by standards. • Largely depend on software. • Less uncertainty (unless some hardware sensors are involved. e.g, data from weather stations.) • Not tangible and more reliable. • Some web services accept data as input and produce some data based on them (e.g. data fusion). • Data send to the consumer using web services. • Comparatively, fewer web services will accessible over the Internet by 2020 [36]. • Typically provide more meaningful processed and refined data.	• Sensors and data processing components compose together into work-flows. • No standards (yet) [12]. • Largely depend on hardware, firmware, as well as software • More uncertainty. • Sensors are Tangible, could be mobile and less reliable. • Some sensors may accept queries/conditions/preferences as inputs and produce data based on them. Nevertheless, sensors do not accept raw data with the intention of fusing data. • Comparatively, more sensors will be accessible over the Internet by 2020 [31]. • Sensors, typically provides less meaningful raw sensor data where they need to be processed by data processing components.

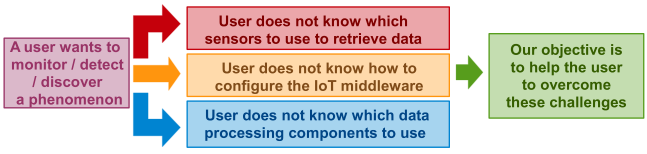

Fig. 3. The problem definition in general.

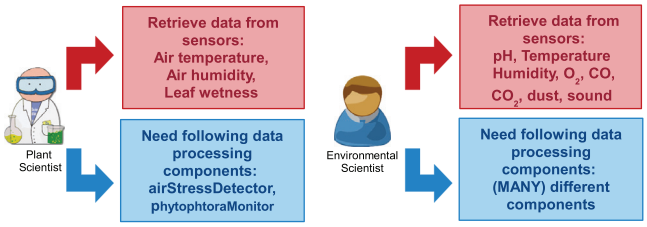

Fig. 4. Use cases that illustrates the need of CASCOM.

experts to configure sensors and data processing mechanisms in an IoT middleware according to their requirements?'. The notations we use in this section are presented in Table 2. Other notations we used in this paper are as follows: Wrapper (W) and Virtual Sensor (VS).

Fig. 4 illustrates two scenarios from two different domains. Each of them has different consumer requirements that lead to two different execution flows. We selected these two scenarios due to the fact that, together, they allow us to showcase the full capabilities of Context-Aware Sensor Configuration Model (CASCOM). In use case 1, a plant scientist wants to monitor whether the experimental crops can be infected by *Phytophtora* [3] disease or not. *Phytophtora* is a fungal disease which can enter a field through a variety of sources. The development and associated attack of the crop depends strongly on the climatological conditions within the field. Humidity plays a major role in the development of *Phytophtora*. Both temperature and whether or not the leaves are wet are also important indicators to monitor *Phytophtora*. The following facts explain *Phytophtora* monitoring (simplified for demonstration purposes). It is important to highlight that rule-based reasoning[4] does not intended to replace rule engines [32]. The objective here is to create the data items that are required by the application.

- IF **Air Temperature** $< \alpha$ AND **Air Humidity** $< \beta$ THEN **Air Stress** level = low ELSE **Air Stress** level = high
- IF **Air Stress** = high AND **Leaf Wetness** $> \delta$ THEN *Phytophtora Disease* = Can-be-infected ELSE = Cannot-be-infected

One of the responsibilities of an IoT middleware is to combine different sensors and data processing components autonomously and produce a data stream as illustrated in Fig. 2. Mostly, our focus is on data streams that consists of multiple data types. A data consumer can feed the data stream into an application for further complex processing such as visualization and modelling that allows the data consumers to achieve their objectives. The main challenge is that the plant scientist may not know (or remember) the domain knowledge listed as rules above. Further, we should not expect a plant scientist to write XML or Java code as part of the configuration. An ideal IoT middleware should help the scientist (non-IT expert) to overcome these challenges by providing tools that are easy to use. The scientist should be able to configure the middleware according to the problems/tasks at hand with minimum effort. Additionally, advanced customization will be useful to optimize the configuration process. Comparatively, use case 1 is less complex as there is only one way to monitor the disease (above rules). For example, the sensor types and data processing components need to be used are straight forward.

- **Use case (1) Solution:** $((S_{AT}, S_{AH}) \Rightarrow C_1, S_{LW}) \Rightarrow C_2$

As symbolized in the above statement, the IoT resource may need to be composed as follows. First, air temperature (S_{AT}) and air humidity (S_{AH}) need to be fed into *airStressDetector* component (C_1). Then, it produces the *airStress* as the outcome. Then leaf wetness (S_{LW}) and airStress need to be fed into *phytophtoraMonitor* component (C_2). It produces the *phytophtoraDisease* status. The IoT resource composition is illustrated in Fig. 5

[4] we employed rules based reasoning in this disuccions.

Table 2
Common algorithmic notations.

Symbol	Definition
$\mathbb{S}$	Complete set of sensors described in the data model.
S_α	S denotes the sensor and subscript α denotes the types of the sensor. Examples are listed in Table 3.
$\mathbb{M}_\theta$	Model represent the complete ontology based semantic data model. The θ can be replaced by either c as (i.e. $\mathbb{M}_c$) or s (i.e. $\mathbb{M}_s$). c denotes the complete model and s denotes the subset of the complete model.
$\mathbb{T}$	Filtered set of tasks described in the data model.
T_u	Task selected by the user (or sensor data consumer) where IoT middleware needs to be configured accordingly.
Ψ_β	SPARQL query that selects different properties from the data model. β can be replaced by t tasks, a answers, q questions, and d data streams.
$\mathbb{Q}$	Filtered set of questions described in the data model (List of questions).
Q_u	Single question selected by the user to answer.
$\mathbb{A}$	Filtered set of answers described in the data model (List of answers).
A_u	Single answer selected by the user.
$C_\gamma(\Delta)$: z	C denotes the data processing components where γ is used an identifier to distinguish each different component. Arguments/ parameters accept by each components are depicted by Δ as set. Δ may accept one or more inputs as denoted by '$\lambda_\#$'. The symbol # denotes the number of the input parameter. The type of each argument is depicted by letters such as x, y (i.e. $\Delta = \{\lambda_1 x, \lambda_2 y\}$). The return value is depicted by letter after ':' symbol. Examples are listed in Table 4.
$\mathbb{H}$	Filtered set of solutions composed by CASCOM which are capable to producing data streams required by the user.
H_u	A single solution composed by CASCOM which are capable to producing data streams required by the user. Solutions is a composition of sensors and data processing components formulated into a certain order.
$\mathbb{D}$	Filtered set of different data-streams that can fulfil user requirements. Data stream is a continuous flow of data which encompasses several data items.
D	A single data stream is composed with number of different data items.
Π_i	This denotes the *ith* data item of a given data stream.
$\mathbb{R}$	Recommendation list that contains information about sensors and data processing components that are not available. Acquire such resources will help to facilitates user requirement in the future.
$\mathbb{P}$	List of all the data items that are available to be captured by the IoT middleware either directly through active wrappers or by combining / composing such data iteams with data processing components. Depending in the complexity of retrieving and generating data items, we categorize them into number of categories.
P	A single data item that is available to be captured from active wrapper.
$\mathbb{M}$	Additional list context information that can be discovered by the users if needed.
σ	Matrix that stores the information about input / outputs of data processing components and data items in $\mathbb{P}$.

Table 3
Subset of sensors.

Sensor	Explanation
S_{AT}	Air Temperature
S_{AH}	Air Humidity
S_{LW}	Leaf Wetness
S_{CM}	Carbon Monoxide
S_{CD}	Carbon Dioxide
S_{MO}	Molecular Oxygen
S_{ME}	Methane
S_{ND}	Nitrogen Dioxide

Table 4
Subset of DPCs.

Sensor	Explanation
$C_1(\Delta)$: z	airStressDetector
$C_2(\Delta)$: z	phytophtoraMonitor
$C_3(\Delta)$: z	pollutionDetector
$C_4(\Delta)$: z	airQualityMonitor

Configuration becomes a complex task in the use case 2. In this scenario, an environmental scientist wants to measure the environmental pollution in Canberra, Australia. In comparison to the use case 1, there are many different ways to measure and visualize pollution. Different sensors and data processing components can be combined together to fulfil the requirements of data consumers as listed below. Even the same Data Processing Components (DPC) may accept different combination of input in order to perform the same task. DPC is a black box that accept certain types of inputs and produces certain types of outputs. The reasoning happen within a given DPC could be varies from, rules based reasoning, statistical reasoning, logical inferencing machine learning, probabilistic reasoning and so on. As an example, we used a rule based DPC in the paper discussion.

- **Use case (2) Solution 1:** $(S_{CM}, S_{CD}, S_{MO}, S_{ME}, S_{ND}) \Rightarrow C_4$

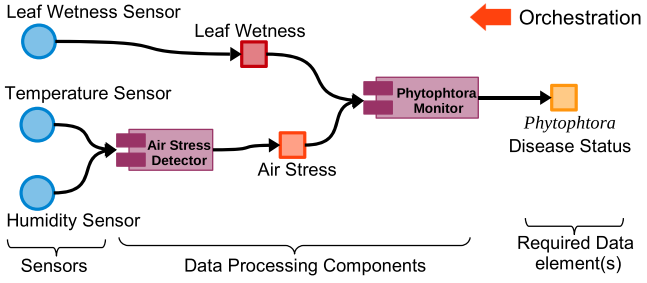

Fig. 5. Resource composition in IoT.

- **Use case (2) Solution 2:** $(S_{CD}, S_{ND}) \Rightarrow C_3$
- **Use case (2) Solution 3:** $(S_{AT}, S_{CD}, S_{ME}) \Rightarrow C_4$

In such circumstances, it is important to consider context information (e.g. accuracy, reliability) and cost of data acquisition (e.g. data communication time and computation time). The availability of more than one option allows a data consumers to make the final decision on which solution to be used depending on the cost and context factors. Both hardware and software costs need to be considered. Additionally, data consumers may need to discover additional context information [25]. Depending on the requirements of the data consumers and application requirements, the required output data stream may vary. Sample data streams, in relation to use case 1, are listed below.

- **Output 1:** airTemperature [double], airHumidity [double], airStress [string],
  leafWetness [double], PhytophtoraDisease [boolean]
- **Output 2:** PhytophtoraDisease [boolean], location [string], batteryLevel[double]

Previously, we explained that the objective of IoT middleware is to produce data streams so the users can inject them into applications. We assume that these applications will accept multiple different data streams as illustrated in Fig. 6. The ideology is

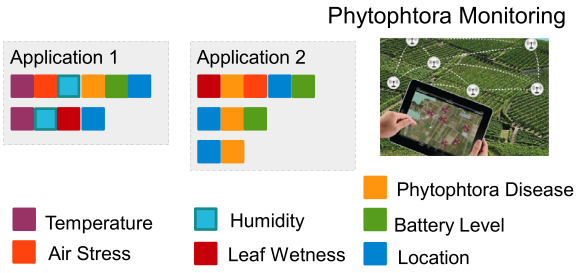

Fig. 6. Each application may accept different data streams and provide outputs at different detail levels.

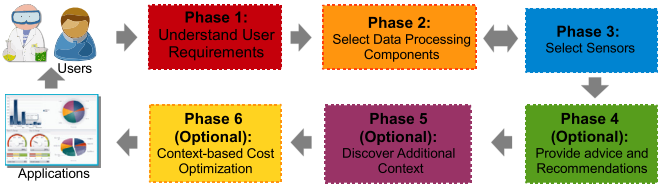

Fig. 7. The Context-Aware Sensor Configuration Model (CASCoM).

that when these applications are provided with more data items, they will perform better or provide additional featured / results. However, each application will have a minimum number of data items that it would accept in order to perform the primary task it promises to deliver. For example, if the application 2 (in Fig. 6) is provided with *PhytophtoraDisease* and *location* data, it will simply mark the areas which are under risk of getting infected by *PhytophtoraDisease*. In contrast, if the application 2 is provided with more information such as *batteryLevel, leafWetness*, and *airStress*, it will produces more detailed and comprehensive visualization that may include risk level with certain confidence. Raw data values of *leafWetness*, and *airStress* may help the application 2 to perform these additional calculations and predictions.

In summary, we assume each application would perform one or more tasks (e.g. *PhytophtoraDisease* monitoring). Each application would accept one more different data streams. In such circumstances, each data stream may consist of different types of data items. Additionally, different developers (or companies) or the same developer may develop multiple applications that perform the same tasks. Similarly, we also assume that there would multiple data processing components that would perform the same data fusion operations though their context information may varied. Due to the large number of possibilities, IoT middleware platforms require an automated process to optimally serve the user requests.

4. Architectural design

Based on the challenges we identified in Section 3, we designed a model, which is supported by a tool, to overcome the difficulties. Context-Aware Sensor Configuration Model (CASCOM) simplifies the IoT middleware configuration process significantly. Our proposed model allows non-technical personnel to configure IoT middleware effortlessly. All the technical configurations are handled internally behind the scenes without the users' involvement. Additionally, we offer several advanced features that allow optimization and customizations. As depicted in Fig. 7, CASCOM consists of six phases. Some phases may or may not be visible to the users. Phases are different from the steps needed to be followed in the CASCOM Tool.

CASCOM Execution flow: In phase 1, data consumers (users) interact with a graphical user interface that is based on a *question-answer (QA)* approach, to specify their requirements. Users can answer as many questions as possible. CASCOM searches and filters the tasks that the user may want to perform. From the filtered list, users can select the desired task. The details of the QA approach are presented later in this section. In phase 2, CASCOM searches for different programming components that allow to generate the data stream required. In phase 3, CASCOM tries to find the sensors that can be used to produce the inputs required by the selected data processing components. If CASCOM fails to produce the data streams required by the users due to insufficient resources (i.e. unavailability of the sensors), it will provide advice and recommendations on future sensor deployments in phase 4. Phase 5 allows the users to capture additional context information. The additional context information that can be derived using available resources and knowledge are listed to be selected. In phase 6, users are provided with one or more solutions.[5] CASCOM calculates the costs for each solution. By default, CASCOM will select the solution with lowest cost. However, users can select the cost models (discussed later in this section) as they required. Finally, CASCOM generates all the configuration files and program codes which the actual IoT middleware may requires[27]. Data starts streaming soon after.

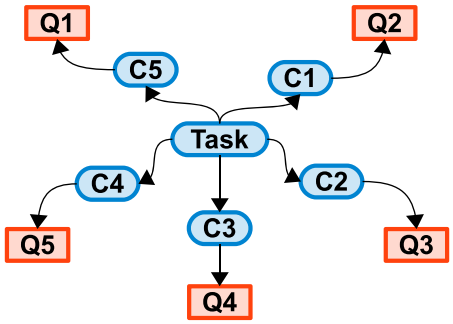

Fig. 8. A part of QA-TDO shows how we developed the QA model. It is important to note the pattern (i.e. *Task* → *Concept* → *Question*).

Phase 1: Understand user requirements: The objective of this phase is to help data consumers to search for a task (e.g. *PhytophtoraDisease* monitoring) that they need to perform easily from a large number of possibilities. For example, data consumers are allowed to narrow down the possibilities by mentioning facts such as domain (e.g, agriculture), and type of the task (e.g. event, visualization). In order to increase the usability, CASCOM retrieves the facts from the data consumers through a QA model (Sample questions: Do you want to visualize data?, Do you want to detect an event?, Do you want to monitor a disease infection? What is the domain your task is related to?). When a user answer a question, the remaining questions will be dynamically selected based on the previous answer. An extract of the proposed *Question and Answer oriented Task Description Ontology* (QA+TDO) is presented in Fig. 8. In QA+TDO, tasks can be explained by any concept as depicted in *C*1, *C*2, etc. in Fig. 8. Each concept should have a *'hasQuestion'* property which links to a question (i.e. *Q*1, *Q*2 and so on). It is expected that new questions will be added to the QA+TDO over time by different domain experts and contributors as part of the knowledge modelling process so the non-technical users can take advantage of them. In QA+TDO, *C* are answers to the questions. (e.g, If *Q*1= What is the domain your task is related to?, then *C*5 is *'domain'* and an individual of *C*5 can be *'agriculture'*.). This process is presented in algorithmic perspective in Algorithm 1, which takes the data model as the input and outputs the user preferred task. The design philosophy of this algorithm is that it repeatedly allow the user to select questions and answer them.

[5] A solution is a combination of sensors and data processing components that can be composed together in order to satisfy the user requirements.

Algorithm 1 Question-answer based task filtering.

Require: ($\mathbb{M}$).
1: **Output:** T_u
2: $\mathbb{M} \leftarrow$ Load data into model
3: **while** T_u != NULL **do**
4: $\mathbb{Q} \leftarrow executeQuery(\Psi_q, \mathbb{M})$
5: $Q_u \leftarrow$ Ask user to select a question from $\mathbb{Q}$
6: Add Q_u to Ψ
7: $\mathbb{A} \leftarrow executeQuery(\Psi_a, \mathbb{M})$
8: $A_u \leftarrow$ Ask user to select a answer from $\mathbb{A}$
9: Add A_u to Ψ
10: $\mathbb{T} \leftarrow executeQuery(\Psi_t, \mathbb{M})$
11: $T_u \leftarrow$ Ask user to select a task from $\mathbb{T}$
12: **if** T_u != NULL **then**
13: **return** T_u
14: **end if**
15: **end while**

As a result, the algorithm will generate a SPAQRL statement and update it every time when a user selects and answers a question. Every time a user answers a question, the number of possible options offered to the users will get reduced as the new Q&A will always add more constraints to the query.

Let us briefly explain the Algorithm 1. CASCOM first allows users to select a question as demoted in Q_u. Next CASCOM uses Q_u to query its knowledge-base. The results are denoted as $\mathbb{A}$. Next, both question and answers is amended to a single query Ψ. Ψ is used to query the knowledge-base and resulted tasks are denoted by $\mathbb{T}$. This process repeats until users find the tasks they are looking for T_u.

Phase 2 and 3: Select sensors and data processing components: CASCOM requires all the information related to sensors and data processing components to be stored in a repository. We extended the Software Component Ontology [9] (SCO) as presented in Fig. 9 in order to model information about data processing components. Further, we modelled sensor descriptions using semantic Sensor Ontology (SSN) [10]. In this phase, the software components are selected in such a way that they can together produce the data stream required to perform the task selected in phase 1. For example, in order to monitor *PhytophtoraDisease*, first CASCOM searches for a software component that can be used to produce the required data. It first finds *PhytophtoraDisease Detector*. The inputs it requires are *air stress* and *leaf wetness*. Phase 3 selects the sensors that produce the output that matches the inputs of the selected component. *Leaf wetness* can be measured directly using hardware sensors. However, *air stress* cannot be detected using any physical sensor. This requires CASCOM to execute phase 2 again in order to find a software component that produces *air stress*. Then CASCOM finds *Air Stress Detector* which takes *air temperature* and *air humidity* as inputs and produces *air stress* as the output. Further, *air temperature* and *air humidity* can be sensed directly through hardware sensors. The IoT middleware configuration process will be completed once the required sensors and data processing components are identified. The remaining phases are optional.

CASCOM performs validation as illustrated in Fig. 10. During the sensors and data processing components composition process, different criteria are evaluated (e.g. data types: int, boolean / measurement units: Celsius, Fahrenheit) in order to verify whether the inputs and outputs are compatible. The above mentioned procedures are presented in algorithmic perspective in Algorithm 2.

This algorithms takes the data model (demoted by $\mathbb{M}$) and the data stream elements of the user preferred task as the inputs (demoted by T_u). First, it finds out what are the output data stream required by the user preferred task (denoted by $\mathbb{D}$). Then, it at-

Algorithm 2 Resources composition and recommendation.

Require: ($\mathbb{M}$), (T_u)
1: **Output:** $\mathbb{H}, \mathbb{R}$
2: $\mathbb{M} \leftarrow$ Load data into model
3: $\mathbb{D} \leftarrow executeQuery(\Psi_d, T_u, \mathbb{M})$
4: $//D = \{\Pi_1, \Pi_2, \Pi_3 ... \Pi_n\}$
5: **for all** $D \in \mathbb{D}$ **do**
6: **for all** $\Pi_i \in D$ **do**
7: **if** $\Pi_i ==$ output of a sensor S_α in the set $\mathbb{S}$ **then**
8: add S_α to H_u
9: **else**
10: **if** $\Pi_i ==$ output of a component C_γ in the set $\mathbb{C}$ **then**
11: add C_γ to H_u
12: $composeFurther(C_\gamma, H_u)$
13: **else**
14: add $\lambda_\#$ to $\mathbb{R}$
15: **end if**
16: **end if**
17: **end for**
18: **end for**
19:
20: **Function** $composeFurther(C_\gamma, H_u)$
21: **for all** $\lambda_\# \in \Delta$ **of** C_γ **do**
22: **if** $\lambda_\# ==$ output of a sensor S_α in the set $\mathbb{S}$ **then**
23: add S_α to H_u
24: *break*
25: **else**
26: **if** $\Pi_i ==$ output of a component C_γ in the set $\mathbb{C}$ **then**
27: add C_γ to H_u
28: $composeFurther(C_\gamma, H_u)$
29: **else**
30: add $\lambda_\#$ to $\mathbb{R}$
31: **end if**
32: **end if**
33: **end for**

tempts to generate that data stream by composing sensors and data processing components (from line 5–18). The design philosophy behind the search and composition is that priority is given to prepare the output data stream using direct sensor outputs (denoted by S_α). If this is not possible (e.g. when a certain data element cannot be directly sensed), the algorithm will search for a data processing component which may be able to produce the required output (denoted by C_γ). If it succeeds, then the inputs of the selected data processing component will be searched (using the $composeFurther(C_\gamma, H_u)$). As illustrated in Fig. 5, this process will continue until the algorithms finds ways to produce the elements in the required output data stream.

Phase 4 (Optional): Provide advice and recommendations: Through comparing SSN ontology and SCO, this phase identities the resource insufficiencies and provides advice to the data consumers regarding future sensor deployments and software component acquisition. This phase provides alternative advice if there are multiple ways to address the insufficiencies (e.g. use case 2). As presented in Algorithm 2, resource insufficiencies are also detected and identified during the resource composition process. A list of resource insufficiencies is prepared and returned as $\mathbb{R}$.

Lets consider use case 2. Its objective is to determine environmental pollution in a city. As presented in Section 3, there are three different solutions that that can be used to achieve this objective. Assume, in our IoT system, we only have access to sensors S_{CD} and S_{ME} . However, those two sensors are not capable of producing data that is required by any of the exiting DPCs, namely C_3 and C_4. Therefore, this phase of our model recommends users to

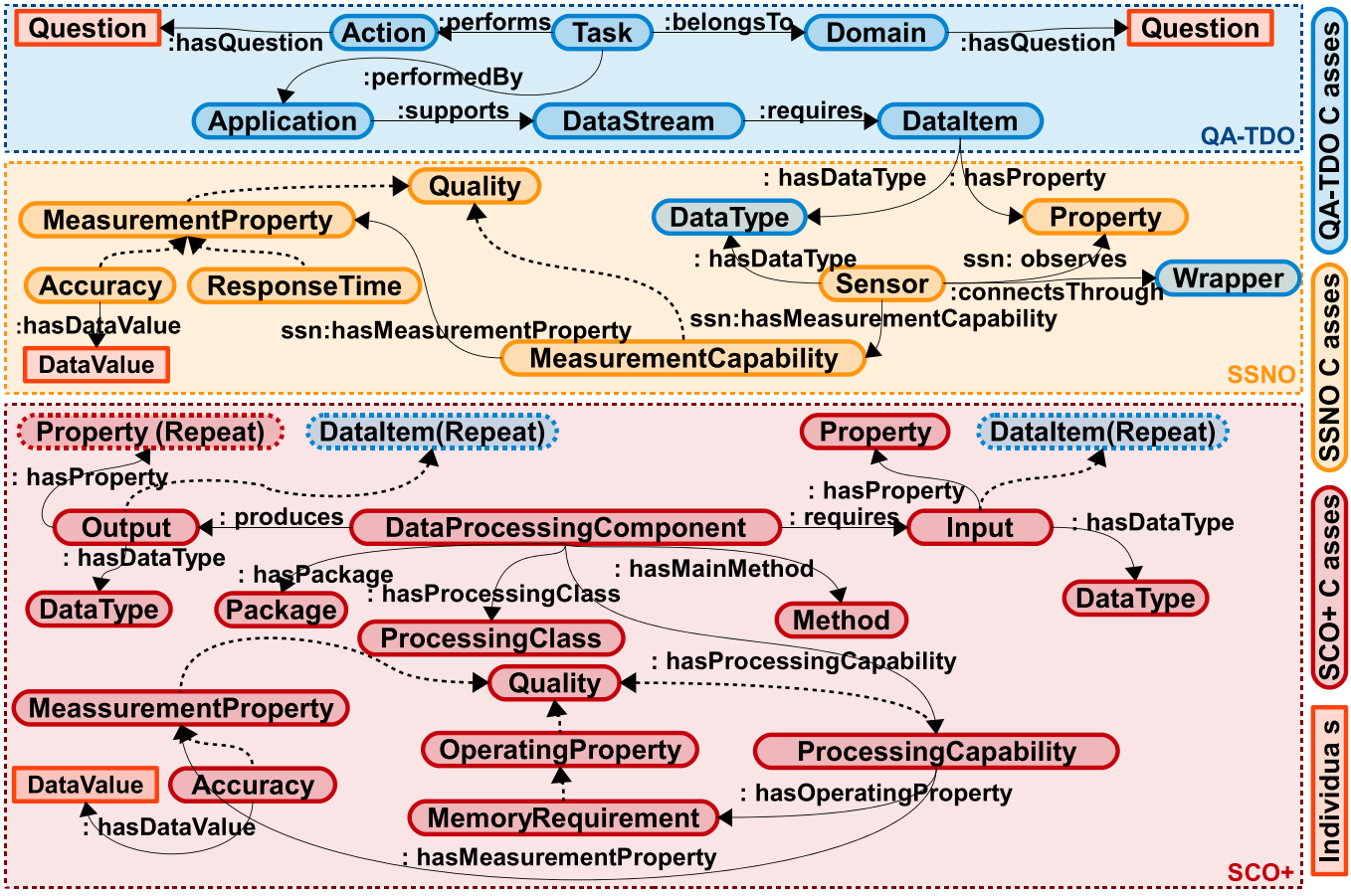

Fig. 9. Extracts of different ontological data models used in CASCOM: QA-TDO, SCO [9], and SSN ontology [10]. The colour coding refers to different prefixes. Prefixes in abbreviated Internationalized Resource Identifiers (IRIs).

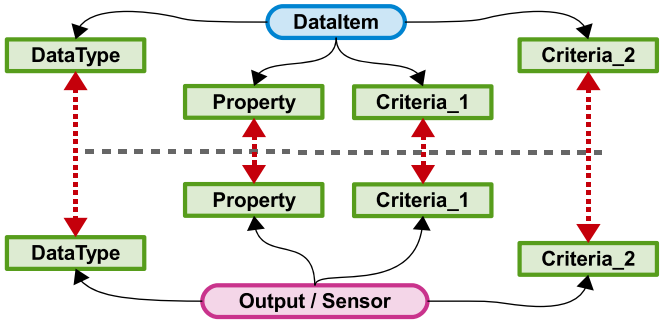

Fig. 10. IoT Resource compositions need to be validated before presented to the users. Semantic meanings as well as syntactic definition (e.g. programming level data types) need to matched and compatible.

deploy either sensors S_{ND} or S_{AT}. Such deployments will fulfil the data requirements of above components.

Phase 5 (Optional): Additional context discovery: With the help of knowledge modelled in ontologies, this phase discovers context information that can be derived by using sensor data. Additional context information such as sensor location and sensor battery life may be required by applications in order to perform complex tasks such as geographical visualization and developing energy-aware sensing schedules. Therefore, discovering additional context is important. Each application may have a compulsory set of inputs that it needs to perform the primary task, though they may accept additional context information in order to provide enhanced results.

First, all the data items directly retrieved through sensors (we call them parameters and denoted by $\mathbb{P}$) are added to a list of context information denoted by $\mathbb{M}$. Such pieces of context information are referred to as primary context [25]. Each wrapper has set of data items (i.e. parameters) it can produce. Set of parameter produce by each active wrapper is noted by $\{\mathbb{P}^0, \mathbb{P}^1, \mathbb{P}^2 \ldots \mathbb{P}^n\}$. Next, these primary context parameters are composed with all existing DPCs (denoted by $\mathbb{C}$) to check whether secondary context [25] can be produced. If possible, such secondary context parameters are also added to the list of context information denoted $\mathbb{M}$ as well. The context discovery procedures are presented in algorithmic perspective in self-explanatory Algorithm 3. Further, We can explain the context discovery procedure using the Fig. 11. This algorithm works independently from users' preferences. Further, it can also be preprocessed. The design philosophy behind this algorithm is that it attempts to identify all possible secondary context information that can be generated by combining all the possible outputs of sensors as well as data processing components. This algorithm is only required to run when a new sensor or data processing component appears or when existing resource disappears.

Phase 6 (Optional): Context-based cost calculation: In CASCOM, the main objective is to identify the required IoT resources at a conceptual-level. In the first 5 phases, we achieve this main objective. In phase 6, we focus on identifying actual IoT resources. It is important to note that there can be multiple DPCs that can perform similar tasks. Further, there are large numbers of sensors

Algorithm 3 Context discovery.

Require: (List of Active Wrappers).
Output: $\mathbb{M}$
$\mathbb{P} = \{\mathbb{P}^0, \mathbb{P}^1, \mathbb{P}^2 ... \mathbb{P}^n\}$
$\mathbb{P}^0 \leftarrow$ List data items available through active wrappers
add $\mathbb{P}^0$ to $\mathbb{M}$
for all $\mathbb{P}^i \in \mathbb{P}^n$ **do**
  for all $C \in \mathbb{C}$ **do**
    for all $\lambda_{\#} \in C$ **do**
      if $\lambda_{\#} ==$ any P in the set $\mathbb{P}^i$ **then**
        add $\checkmark$ to $\lambda_{\#}$ of C in σ
      end if
    end for
  end for
  for all $C \in \mathbb{C}$ **do**
    if All $\lambda_{\#}$ of $C == \checkmark$ **then**
      add output of C to $\mathbb{P}^{i+1}$
      add $\checkmark$ to $\lambda_{\#}$ of C in σ
    end if
  end for
end for
for all $C \in \mathbb{C}$ **do**
  if All $\lambda_{\#}$ of $C == \checkmark$ **then**
    add output of C to $\mathbb{M}$
  end if
end for

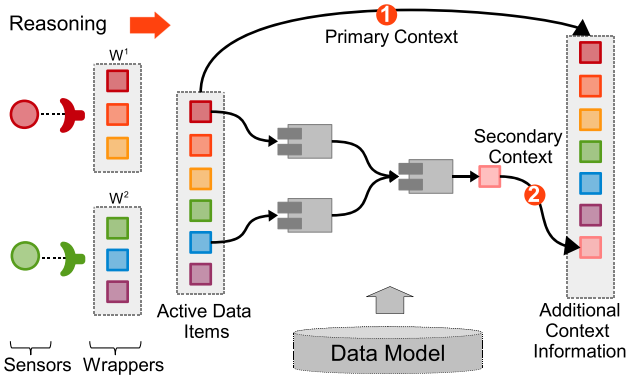

Fig. 11. Primary and secondary context discovery.

available with overlapping and sometimes redundant functionality. In such situation, data consumers may want to decide the exact criteria that IoT resources selection process should consider.

CASCOM performs ontological reasoning to find out all possible solutions. Each solution may combine different sensors and data processing components where their costs may different. For example, different types of sensors can be used to monitor environmental pollution as illustrated in Fig. 4. Cost does not always refer to financial terms (e.g, sensors: energy, bandwidth, latency; data processing: memory requirement, processing time). By default, all the context parameters are treated equally. However, users can define their priorities for each context property in comparative fashion [23]. If the users want more *reliable* sensors, the *reliability* can be defined with more priority, but it may increase the cost.

5. Description generation tool

In our proposed model, the description of IoT resources and related knowledge play a significant role. Today, even though there are sophisticated tools that can be used to develop *ontologies* and model *instances* such as *Protege* [20], they are very complex to use. The learning curve of these tools are significant. The user interface of *Protege* tool, with CASCOM data model opened, is presented in Fig. 12(a). As it is clearly visible, the *Protege* user interface looks very complex to someone who has never used it before and hard to understand where to even begin.

In CASCOM, we expect data processing components, sensors, and domain knowledge to be collectively described and modelled by developers, domain experts, and non-technical personnel (e.g. capabilities, inputs, output, etc.). However, not even all developers are familiar with semantic modelling tools such as *Protege*. Therefore, we built a very simple form-based tool where anyone can learn and use it with very limited effort. They can fill the form and the tool will model the according to CASCOM ontology behind the scenes. More importantly, this tool can be used to add IoT resource descriptions to existing knowledge models. Our form-based knowledge modelling tool in presented in Fig. 12(b). This tool consists of number to separate tabs where each tab allows users to model certain type of knowledge (e.g. describe a data processing component, describe sensors, add domain knowledge, etc.).

6. Implementation and experimentation

This section presents implementation details of our proof of concept development and evaluation from both computational and usability perspectives.

6.1. Testbed

For proof of concept deployment and evaluation, we used a computer with Intel(R) Core i7 CPU and 16GB RAM. We used the Java programming language to develop the CASCOM tool and employed the open source Apache Jena API to manipulate semantic data. We used a Jena TDB-backed[6] approach to store the data. The user interface has been developed using the Java Swing framework. We modelled sensor descriptions according to the Semantic Sensor Network Ontology (SSN) [10]. Further, we modelled data processing components (DPC) descriptions according to the Software Component Ontology Plus (SCO+). The proposed SCO+ is based on SCO [9], but additionally supports modelling context information such as accuracy and reliability as presented in Fig. 9.

To evaluate the proposed model, we developed a software tool that is illustrated in Fig. 13. First, data consumers can select a question that they can answer from the drop down box. Then, they are allowed to answer the question. Possible answers will be listed in the next panel. Next, consumers can either answer another question by clicking *Answer More* button. In contrast, they can click *Search Tasks* button to search possible sensing tasks. Possible sensing tasks will be listed at the bottom of the next panel. Data consumers can select the sensing task they want and click *Search Solution* button. CASCOM will automatically generate different compositions of IoT resource that can perform the sensing task requested by the consumer.

6.2. Methodology

We evaluated CASCOM using both qualitative and quantitative methods. We analysed and compared our proposed solution with respect to the existing GSN configuration model [1]. First, let us present our quantitative evaluation strategy (i.e. computational performance). In Fig. 14a, we examined the feasibility of CASCOM model in term of how much data storage capacity is required as the knowledge-base grows. In Fig. 14b, we examined the feasibility of CASCOM by measuring the variability of the data model loading time as the knowledge-base grows. Next, in Fig. 14c, we evaluated

[6] jena.apache.org/documentation/tdb.

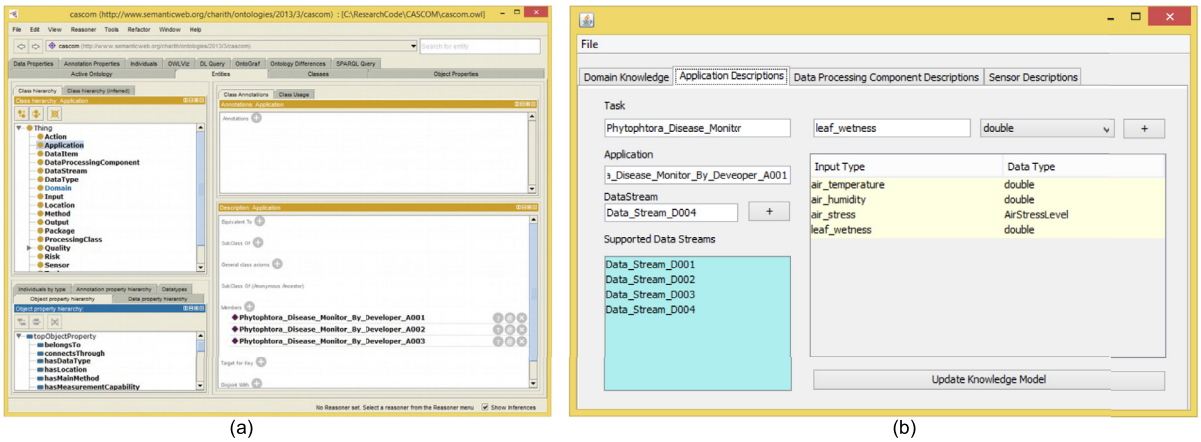

Fig. 12. Semantic Data Modelling Tools: (a) *Protege* and (b) proposed IoT resource description tool.

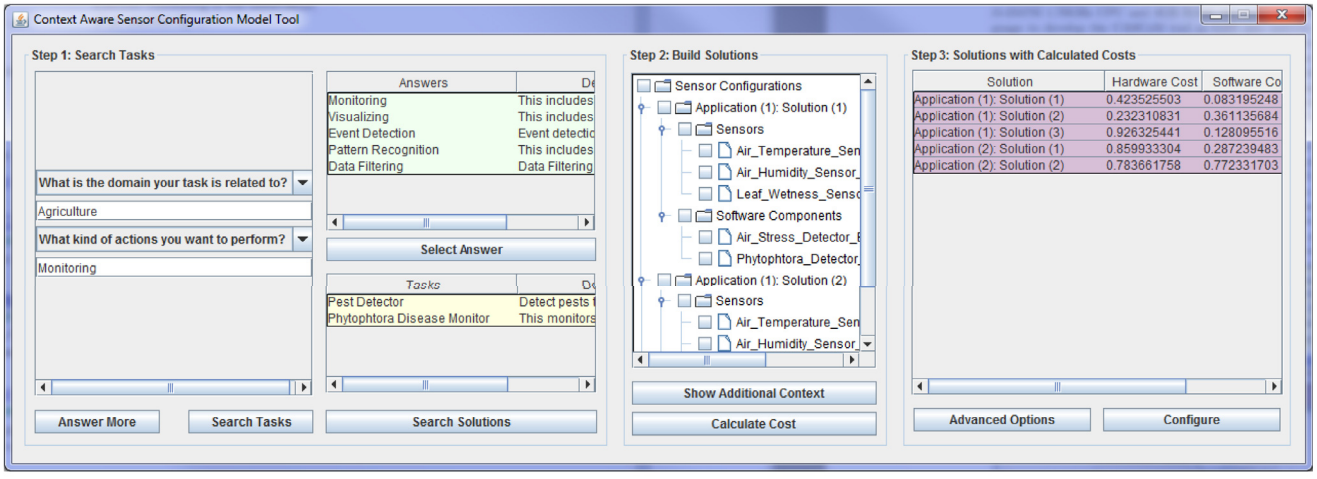

Fig. 13. User interface of the software tool that supports CASCOM.

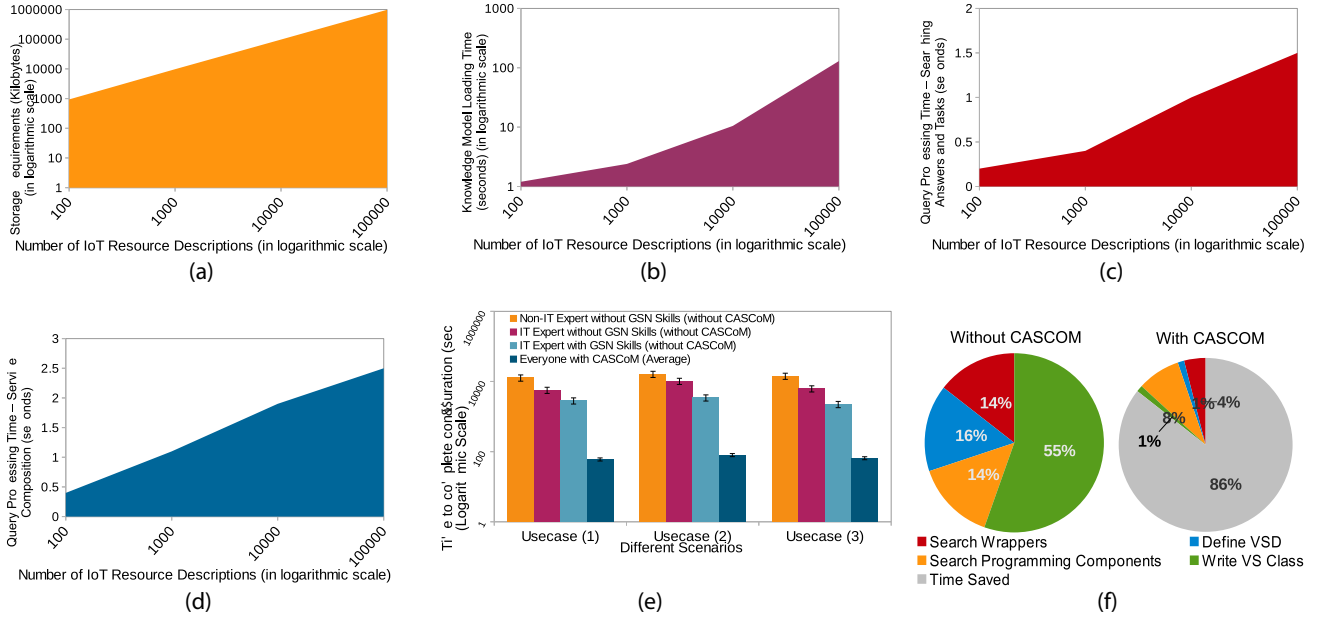

Fig. 14. CASCOM performance evaluations.

the variability of query processing time, related to searching tasks, when the knowledge-base grows. In Fig. 14d, we examined how IoT resource composition and secondary context discovery query processing time varied as the knowledge-base grows. In order to evaluate CASCOM's computational performance, we generated a data model, according to the ontology presented in Fig. 9, that consists of large amount of IoT resource descriptions.

Now, let us present our qualitative evaluation strategy (i.e usability). We used three use-case scenarios for this evaluation. In each use case, the user was required to configure the IoT middleware in such a way that it produces a specific data stream: (1) *monitor Phytophtora disease*, (2) *monitor environmental pollution*, and (3) *monitor and analyse crowd movement (indoor)*. Further, we selected three types of users: (1) *IT experts who were familiar with GSN configuration process*, (2) *IT experts who was were familiar with the GSN*, and (3) *non-IT experts*.

For the usability study, we created a similar data model, but with a small number of IoT resource descriptions. Details of these data sets are presented below. For each use case, a set of basic instructions and programming guidelines that explains the GSN configuration process were given. First, we asked the users to configure the GSN middleware without the support of CASCOM. Secondly, we asked the users to configure the GSN middleware by using CASCOM. We measured the time taken by each user and results are presented in Fig. 14e. In this evaluation, we considered the time taken by both users as well as by the computer to perform resource selection and composition. Further, 31 participants (15 IT expert who were not familiar with the GSN, 15 non-IT experts, and 1 IT expert who was familiar with GSN configuration process) were involved in this experiment. In Fig. 14f, we analysed different phases of the configuration process separately and compared the current approaches with the CASCOM approach. In order to make the results comparable, we assumed the users are IT experts who know the GSN configuration process.

7. Results, discussion and lessons learned

As shown in Fig. 14a, the storage requirement grows linearly[7] when the knowledge-base grows. In semantic modelling, the data model loading time is proportionate to the data model size. Therefore, as expected, loading time also grows linearly when the knowledge-base grows as shown in Fig. 14b. However, it is important to note that the actual data model size and the actual loading time vary based on the data modelling technique used (e.g. file-based, database-based) and the semantic framework employed [6] (e.g. Jena, Sesame). The amount of time that is required to load the CASCOM data model into memory is less than 200 s even when it contains 100,000[8] descriptions. Similarly, Jena-TDB takes only 1GB to model and store 100,000 IoT resource descriptions.[9] In similar conditions, task searching query can return the results in less than 1.5 s as show in Fig. 14c. Further, resource composition can also be completed in 2.5 s as shown in Fig. 14d. When we consider real world deployments, it is very unlikely that a single instance os GSN middleware would host over a 100,000 sensors and data processing components connect to it. Based on these results we can conclude that CASCOM is feasible to use in real-world deployment.

In semantic data modelling, model size, storage requirement, and query times depend on the number of descriptions that are modelled in a given store. Let us consider the data model depicted in Fig. 9. In this model (as well as in our simulations), we have used only a part of the SSN ontology, because the other parts are irrelevant for the composition process. However, if we want to model using the full SSNO, the model size would grow depending on how much more information (i.e. nodes and edges) that we want to include in order to describe a given set of IoT resources.

As also presented in Fig. 14e, non-IT experts required extremely detailed guidelines (compared to IT experts) to perform the configuration as they are not familiar with the activities such as programming. They also required direct verbal assistant from the authors. In addition, it was revealed that non-IT experts and IT experts who are not familiar with GSN were unable to configure the GSN at all without guidelines. In contrast, simple guidelines that explain the GUI allowed all users to complete the given task, using CASCOM, within a fairly similar amount of time. Though the complexity of the user requirement (i.e. configuration related to each scenario) makes visible impact on configuration time in the current GSN approach, it diminishes when users use CASCOM to configure GSN. Fig. 14e shows that CASCOM allows to considerably reduce the time required for configuration of data processing mechanism in IoT middleware. Specifically, CASCOM allowed the three types of users to complete the given task approximately 40, 110 and 210 times faster (respectively) in comparison to the existing approach.

According to Fig. 14f, even IT experts who know GSN can save time by using CASCOM up to 86%. Specially, time taken for defining the Virtual Sensor Definition (VSD) and Virtual Sensor (VS) class have been significantly reduced.[10] Both files can be generated by CASCOM autonomously within a second even for complex scenarios. However, the time taken to find data processing components and sensors (and wrappers) depends on the size of the semantic data model.

As CASCOM models knowledge according to ontologies, users do not need to memorise domain knowledge (i.e. *which sensor data types are required to perform a certain task?*). This is an significant improvement over the existing approach. Due to the employment of semantic technologies, CASCOM is extensible into any domain. More importantly, adding new sensor descriptions and data processing component descriptions to the data model overtime allows CASCOM to compose new solutions. Ontological reasoning allows to deal with inconsistent usage of domain specific terminologies among domain experts. Ontologies helped in CASCOM to deal with performing validating task in composition of data components. Alternative to ontologies, we could have used a configuration file that explains which programming components and sensors need to be used to produce the required data stream for a given application (e.g. template-base approach). However, such an approach will drastically reduce the interoperability and flexibility. In IoT, ideal approaches should be able to dynamically compose and configure sensors and data processing components as it is impossible predict their availability at give time (new sensors and data processing components may available to use).

7.1. Revisiting challenges

In this section, we summarise how the challenges and drawbacks identified in the related work section are being addressed by our proposed solution. Main weakness in the existing solutions, such as Microsoft *SensorMap* [18] and *Linked Sensor Middleware* (LSM) [29], is that they are user driven and not scalable and . Users are expected to conduct discovery and composition by themselves either using naked eye (i.e. looking at the user interface provided) or limited keyword-based search facilities. In contrast, CASCOM is

[7] Graphs in logarithmic scale.

[8] 100,000 means we have modelled 100,000 sensor descriptions, data processing components, and tasks related knowledge descriptions each.

[9] In our synthetic data generation process, we assume each data processing component accepts three inputs and produce one output.

[10] VSD, VS are both configuration files that need to be dealt with when configuring GSN middleware. More details are available in [1].

a knowledge driven approach where users only required to input very high level user requirement. The discovery and composition is done autonomously based on the knowledge model. As CASCOM models knowledge according to ontologies, users do not need to memorise domain knowledge. This is an significant improvement over the existing approach such as Microsoft *SensorMap* and *Linked Sensor Middleware* (LSM). Our results also show that knowledge driven approach allowed users to accomplish their task much faster than the user driven approaches.

Additionally, we successfully demonstrated how hardware resources and software resources can be composed into work flows to achieve certain tasks. This is an advancement over existing approaches such as web service composition [14,17] where only software services are composed together.

8. Conclusions and future work

In this paper, we proposed a semantic knowledge driven IoT resource discovery and composition engine to assist sensor data consumers to retrieve the data they want quickly and effortlessly. In particular, we focus on facilitating non-technical users to use IoT middleware platforms without spending too much time on learning technical details. To achieve this, we developed an IoT middleware configuration model called CASCOM. CASCOM makes the configuration process much easier by providing a sophisticate graphical user interface to express user requirements. Through a proof of concept implementation, we evaluated CASCOM both in term of usability and computational complexity. The results shows that the proposed model is significantly useful for non-technical personal to use IoT middleware platforms to retrieve data. CASCOM engine is highly flexible and scalable due to its knowledge driven nature where we can add more descriptions about data processing components and sensors over time. We have also done computational evaluations to demonstrate the feasibility and scalability of our proposed model. In additions to its primary role, CASCOM is capable of discovering secondary context through processing primary context information.

In the future, we would like to incorporate privacy aspects into the model. Currently, CASCOM is not considering any privacy violations that may occur when data processing components and sensors are composed together. It is important to evaluate and verify all consumer requests received by an IoT middleware to make sure that data owners' privacy are protected at all times. More importantly, for some consumer tasks (e.g. monitor and analyse crowd movement (indoor)), privacy would be a greater concern than for others (e.g. monitor *Phytophtora* disease).

Acknowledgments

Dr. Charith Perera's work has been funded by The Australian National University, The Commonwealth Scientific and Industrial Research Organisation (CSIRO), and European Research Council Advanced Grant 291652 (ASAP).

References

[1] K. Aberer, M. Hauswirth, A. Salehi, Infrastructure for data processing in large-scale interconnected sensor networks, in: International Conference on Mobile Data Management, 2007, pp. 198–205.
[2] L. Atzori, A. Iera, G. Morabito, The internet of things: A survey, Comput. Netw. 54 (15) (2010) 2787–2805.
[3] A. Baggio, Wireless Sensor Networks in Precision Agriculture, Technical Report, Delft University of Technology The Netherlands, 2009. http://www.sics.se/realwsn05/papers/baggio05wireless.pdf [Accessed on: 2012-05-10].
[4] S. Bandyopadhyay, M. Sengupta, S. Maiti, S. Dutta, Role of middleware for internet of things: A study, Int. J. Comput. Sci. Eng. Surv. 2 (2011) 94–105.
[5] A. Bassi, M. Bauer, M. Fiedler, T. Kramp, R. Kranenburg, S. Lange, S. Meissner (Eds.), Enabling Things to Talk: Designing IoT solutions with the IoT Architectural Reference Model, Database Management & Information Retrieval, Springer-Verlag, Berlin Heidelberg, 2013.
[6] C. Bizer, A. Schultz, The berlin sparql benchmark, Int. J. Semantic Web Inf. Syst. 5 (2) (2009) 1–24.
[7] J. Bronsted, K. Hansen, M. Ingstrup, Service composition issues in pervasive computing, Pervas. Comput. IEEE 9 (1) (2010) 62–70.
[8] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, I. Brandic, Cloud computing and emerging {IT} platforms: Vision, hype, and reality for delivering computing as the 5th utility, Future Generat. Comput. Syst. 25 (6) (2009) 599–616.
[9] F.E. Castillo-Barrera, R.C.M. Ramirez, H.A. Duran-Limon, Knowledge capitalization in a component-based software factory: a semantic viewpoint, in: LA-NMR, 2011, pp. 105–114.
[10] M. Compton, P. Barnaghi, L. Bermudez, R. Garca-Castro, O. Corcho, S. Cox, J. Graybeal, M. Hauswirth, C. Henson, A. Herzog, V. Huang, K. Janowicz, W.D. Kelsey, D.L. Phuoc, L. Lefort, M. Leggieri, H. Neuhaus, A. Nikolov, K. Page, A. Passant, A. Sheth, K. Taylor, The ssn ontology of the w3c semantic sensor network incubator group, Web Semantics 17 (0) (2012) 25–32.
[11] J. El Hadad, M. Manouvrier, M. Rukoz, Tqos: Transactional and qos-aware selection algorithm for automatic web service composition, Serv. Comput. IEEE Trans. 3 (1) (2010) 73–85.
[12] J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, Internet of things (iot): A vision, architectural elements, and future directions, Future Generat. Comput. Syst. 29 (7) (2013) 1645–1660. Including Special sections: Cyber-enabled Distributed Computing for Ubiquitous Cloud and Network Services & Cloud Computing and Scientific Applications Big Data, Scalable Analytics, and Beyond.
[13] S. Hodges, S. Taylor, N. Villar, J. Scott, D. Bial, P. Fischer, Prototyping connected devices for the internet of things, Computer 46 (2) (2013) 26–34.
[14] S. Khemakhem, K. Drira, M. Jmaiel, Semantic matching to achieve software component discovery and composition, Technical Report, Laboratory for Analysis and Architecture of Systems, 2012. http://hal.archives-ouvertes.fr/docs/00/79/62/46/PDF/paper12.pdf [Accessed on: 2013-02-05].
[15] K. Kritikos, D. Plexousakis, Requirements for qos-based web service description and discovery, IEEE Trans. Serv. Comput. 2 (4) (2009) 320–337.
[16] P. Leitner, W. Hummer, S. Dustdar, Cost-based optimization of service compositions, Serv. Comput. IEEE Trans. 6 (2) (2013) 239–251.
[17] E. Maximilien, M. Singh, A framework and ontology for dynamic web services selection, Internet Comput. IEEE 8 (5) (2004) 84–93.
[18] S. Nath, J. Liu, F. Zhao, Sensormap for wide-area sensor webs, Computer 40 (7) (2007) 90–93.
[19] H. Noguchi, T. Mori, T. Sato, Automatic generation and connection of program components based on rdf sensor description in network middleware, in: Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on, 2006, pp. 2008–2014.
[20] N. Noy, M. Sintek, S. Decker, M. Crubezy, R. Fergerson, M. Musen, Creating semantic web contents with protege-2000, Intel. Syst. IEEE 16 (2) (2001) 60–71.
[21] C. Pahl, An ontology for software component matching, in: Proceedings of the 6th International Conference on Fundamental Approaches to Software Engineering, in: FASE'03, Springer-Verlag, Berlin, Heidelberg, 2003, pp. 6–21.
[22] S. Patidar, D. Rane, P. Jain, A survey paper on cloud computing, in: Advanced Computing Communication Technologies (ACCT), 2012 Second International Conference on, 2012, pp. 394–398.
[23] C. Perera, A. Zaslavsky, P. Christen, M. Compton, D. Georgakopoulos, Context-aware sensor search, selection and ranking model for internet of things middleware, in: IEEE 14th International Conference on Mobile Data Management (MDM), Milan, Italy, 2013, pp. 314–322.
[24] C. Perera, A. Zaslavsky, P. Christen, D. Georgakopoulos, Ca4iot: Context awareness for internet of things, in: IEEE International Conference on Conference on Internet of Things (iThing), Besanon, France, 2012, pp. 775–782.
[25] C. Perera, A. Zaslavsky, P. Christen, D. Georgakopoulos, Context aware computing for the internet of things: A survey, Commun. Surv. Tut. IEEE 16 (1) (2013) 414–454.
[26] C. Perera, A. Zaslavsky, P. Christen, D. Georgakopoulos, Sensing as a service model for smart cities supported by internet of things, Trans. Emerg. Telecommun. Technol. (ETT) 25 (1) (2014) 81–93.
[27] C. Perera, A. Zaslavsky, M. Compton, P. Christen, D. Georgakopoulos, Context aware sensor configuration model for internet of things, in: Proceedings of the 12th International Semantic Web Conference (Poster & Demo) (ISWC), Sydney, Australia, 2013, pp. 253–256.
[28] D.L. Phuoc, M. Hauswirth, Linked open data in sensor data mashups, in: In Proceedings of the 2nd International Workshop on Semantic Sensor Networks (SSN09), volume 522, CEUR Workshop at ISWC 2009, Washington DC, USA, 2009, pp. 1–16.
[29] D.L. Phuoc, H.N.M. Quoc, J.X. Parreira, M. Hauswirth, The linked sensor middleware - connecting the real world and the semantic web, in: International Semantic Web Conference (ISWC), 2011.
[30] Z. Song, A. Cá andrdenas, R. Masuoka, Semantic middleware for the internet of things, in: Internet of Things (IOT), 2010, 2010, pp. 1–8.
[31] H. Sundmaeker, P. Guillemin, P. Friess, S. Woelffle, Vision and Challenges for Realising the Internet of Things, Technical Report, European Commission Information Society and Media, 2010. http://www.internet-of-things-research.eu/pdf/IoT_Clusterbook_March_2010.pdf [Accessed on: 2011-10-10].
[32] K. Taylor, L. Leidinger, Ontology-driven complex event processing in heterogeneous sensor networks, in: Proceedings of the 8th Extended Semantic Web Conference on the Semanic Web: Research and Applications - Volume Part II, in: ESWC'11, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 285–299.

[33] T. Teixeira, S. Hachem, V. Issarny, N. Georgantas, Service oriented middleware for the internet of things: A perspective, in: ServiceWave, in: Proceedings of the 4th European conference on Towards a service-based internet, Springer-Verlag, Poznan, Poland, 2011, pp. 220–229.
[34] Q. Wu, Q. Zhu, Transactional and qos-aware dynamic service composition based on ant colony optimization, Future Generat. Comput. Syst. 29 (5) (2013) 1112–1119. Special section: Hybrid Cloud Computing.
[35] A. Zaslavsky, C. Perera, D. Georgakopoulos, Sensing as a service and big data, in: International Conference on Advances in Cloud Computing (ACC-2012), Bangalore, India, 2012, pp. 21–29.
[36] Z. Zheng, Y. Zhang, M. Lyu, Investigating qos of real-world web services, Serv. Comput. IEEE Trans. 7 (1) (2014) 32–39.

Charith Perera is a Research Associate at The Open University, UK. Previously he worked at the Information Engineering Laboratory, ICT Center, Commonwealth Scientific and Industrial Research Organization (CSIRO). Perera received his BSc (Hons) in Computer Science from Staffordshire University, Stoke-on-Trent, UK and MBA in Business Administration from University of Wales, Cardiff, UK and PhD in Computer Science from The Australian National University, Canberra, Australia. His research interests include the Internet of Things, smart cities, mobile and pervasive computing, context-awareness, and ubiquitous computing. He is a member of both IEEE and ACM. Contact him at charith.perera@ieee.org.

Athanasios V. Vasilakos is a professor at Dept of Computer Science, Electrical and Space Engineering, Lulea University of Technology, Lulea, Sweden. He has served as General Chair, and Techinical Program Committee Chair for many internation conferences. He also served or is serving as Editor or/and Guest Editor for many technical journals, such as IEEE TNSM, IEEE TSMC-partB, IEEE TITB, IEEE JSAC special issues of May 2009, Jan. 2011, March 2011, ACM TAAS and IEEE Communications Magazine. He is founding Editor-in-Chief of the International Journal of Adaptive and Autonomous Communications Systems (IJAACS) and the International Journal of Arts and Technology (IJART). He is also General Chair of the Council of Computing and Communications of the European Alliances for Innovation.