Cloud Sensor Ontology and Linked Data to Support Autonomicity in Cloud Application Platforms

Rustem Dautov[1], Iraklis Paraskakis[1], and Mike Stannett[2]

[1] South-East European Research Centre (SEERC),
CITY College — International Faculty of the University of Sheffield,
Thessaloniki, Greece
{rdautov,iparaskakis}@seerc.org
[2] Department of Computer Science, University of Sheffield, UK
m.stannett@sheffield.ac.uk

Abstract. Cloud application platforms with their numerous deployed applications, platform and third-party services are becoming increasingly complex, dynamic and data-intensive, and require novel intelligent approaches to be applied in order to maintain them at an operational level. By treating cloud application platforms as distributed networks of software sensors and utilising techniques from the Semantic Sensor Web area, we have developed a monitoring framework which allows us to detect, diagnose and react to emerging critical situations in complex environments of cloud application platforms in a dynamic manner. In this paper, we focus on our use of a Sensor Cloud Ontology to: (i) represent cloud-based logical software sensors; (ii) homogenise monitored sensor data in the form of RDF streams; and (iii) apply stream and static reasoning to these monitored values in order to detect critical situations. We also explain how utilisation of Linked Data principles can help achieve a more flexible and extensible architecture to define diagnosis and adaptation policies. We discuss benefits associated with our approach, as well as potential shortcomings and challenges.

Keywords: Cloud Computing, Autonomic Computing, Semantic Sensor Web, SSN Ontology, Linked Sensor Data.

1 Introduction and Motivation

Since its emergence nearly 15 years ago [3,4], the Semantic Web stack has developed into a wide range of solutions and technologies whose purpose is no longer limited to providing computer-readable meaning to the Web, but now encompasses a range of problem domains, not necessarily related to the Semantic Web, where existing challenges dictate a need for novel intelligent approaches.

One such area is the domain of Cloud Application Platforms (CAPs). These are a group of Platform-as-a-Service (PaaS) cloud offerings, characterised by extensive customer support for developing, testing, deploying and maintaining software. CAPs not only provision their customers with an operating system and run-time execution environment, but additionally offer a range of generic, reliable, composable and reusable services, following the principles of Service-Oriented Computing (SOC) [22,27]. For example, Google App Engine[1] currently offers 41 services (or "features"), Microsoft Windows Azure[2] provides 17 built-in services and 46 add-ons (i.e., third-party services registered with the platform), and Heroku[3] offers over 100 add-on services.

However, such a flexible model for application development, in which complex application systems are assembled from existing components, has its pitfalls. Cloud platform providers increasingly find themselves in a situation where the ever-growing complexity of resulting environments poses new challenges as to how large volumes of actively streaming, heterogeneous and uncertain data should be dynamically analysed to support situation assessment and run-time adaptations. Accordingly, our research focuses on how Semantic Web technologies (specifically, OWL ontologies, SWRL rules, RDF streams and continuous SPARQL query languages) can be utilised to define semantic streams of monitored data which will then be queried and reasoned over in order to perform situation assessment and suggest further adaptation strategies. Using these technologies has allowed us to develop a small-scale prototype self-adaptation framework which enables dynamic monitoring and intelligent analysis of flowing data within CAPs to support run-time adaptations.

The rest of the paper is organised as follows. Section 2 is dedicated to background information, outlining both the context of the research presented in this paper, and some of our earlier findings. It briefs the reader on: (a) our approach to treating CAPs as distributed networks of software sensors; and (b) the self-adaptation framework for CAPs. In Section 3 we study existing ontologies for modelling cloud environments and sensor-enabled domains, and position our work at the intersection of these two domains. Section 4 describes the Cloud Sensor Ontology which lies at the core of our self-adaptation framework, and illustrates its role in the definition of RDF streams, C-SPARQL queries and SWRL rules using an example based on Heroku add-on services. In Section 5 we elaborate on the presented semantic approach and explain how it can be further extended utilising Linked Data principles. Section 6 concludes the paper.

2 Background

2.1 Cloud Application Platforms as Sensor Networks

A fundamental underpinning of our approach is our interpretation of CAPs as distributed networks of "software sensors" – that is, services, deployed applications, platform components, etc., which constantly emit raw heterogeneous data

[1] `https://cloud.google.com/products/app-engine/`
[2] `http://azure.microsoft.com/`
[3] `https://heroku.com`

which has to be monitored and analysed to support run-time situation assessment.[4] This enables us to apply existing solutions developed by the Semantic Sensor Web (SSW) community, which address the requirements of Sensor Web Enablement [8] by combining ideas from two research areas, the Semantic Web and the Sensor Web; this combination enables situation awareness by providing enhanced meaning for sensor observations [28]. In particular, we were inspired by the Semantic Sensor Networks (SSN) approach to express heterogeneous sensor values in terms of RDF triples using a common ontological vocabulary, and have created our own Cloud Sensor Ontology (CSO) to act as the core element of a self-adaptation framework for CAPs.

2.2 Self-adaptation Framework for CAPs

Fig. 1 demonstrates a high-level architecture of the self-adaptation framework, taking the established MAPE-K model [21] as an underlying model for self-adaptation. In order to support both self-awareness and context-awareness of the managed elements (i.e., software sensors within CAPs), we needed to develop certain modeling techniques to define the adaptation-relevant knowledge of the cloud environment (e.g., platform components, available resources, connections between them, entry-points for monitoring and execution, adaptation and diagnosis policies, etc.). In particular, we wanted to ensure:

– separation of concerns;
– the ability to make flexible modifications through declarative definitions;
– enhanced reuse capabilities, automation and reliability (as opposed to traditional hard-coded approaches).

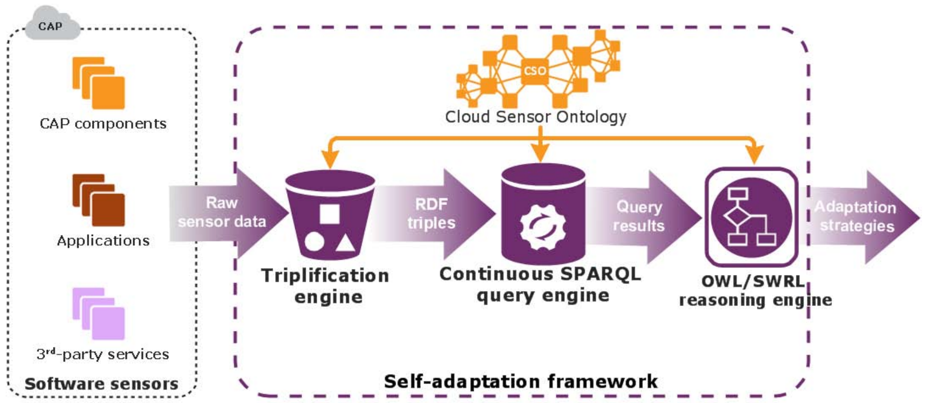

Fig. 1. High-level architecture of the self-adaptation framework

Our solution was to develop a Cloud Sensor Ontology, which also serves as a common vocabulary of terms, shared across the whole managed system, and

[4] For a more detailed overview of our approach we refer interested readers to [12].

corresponds to the Knowledge component of the MAPE-K model. Accordingly, our ontological classes and properties serve as “building blocks” for creating RDF streams, C-SPARQL [2] queries and SWRL rules. By annotating monitored values with semantic descriptions, we enabled the framework to combine observation streams with static ontological knowledge and perform run-time formal reasoning. This work in turn opened promising opportunities for performing run-time analysis, problem diagnosis, and suggesting further adaptation actions [14]. In this paper we focus on the Semantic Web aspects of our approach.

3 Related Work

There has been a considerable amount of research efforts in the direction of conceptually modelling cloud environments with ontologies and thus benefit from declarative definitions, human-readability, built-in reasoning capabilities, standardised languages, interoperability, easy accessibility, etc. [35]. In [1] Androcec et al. provide a holistic view on the existing works and presents a systematic review of 24 cloud ontologies. According to this review, the whole body of work can be classified into four main categories:

– *Cloud resources and services description* – studies in this category describe cloud delivery models (i.e., IaaS, PaaS, SaaS), resources and services, pricing models, etc. Examples of ontologies belonging to this category include [7,15,24,33,37]. However, broadly speaking, all cloud ontologies can be classified under this categor et al.y, since they all describe cloud resources to certain extend.
– *Cloud security* – this category of ontologies looks at clouds from a perspective of modelling security- and privacy-related aspects. For example, in [32] Takahashi et al. devised an ontology based on cyber-security operational information of cloud systems, and developed the Countermeasure Knowledge Base – a set of assessment rules with scoring methodologies and check-lists.
– *Cloud interoperability* – studies in this category use ontologies to achieve interoperability among various cloud providers, their services and APIs (often based on existing standards and proposals for software interoperability), and thus minimise the so-called “lock-in” effect. A notable example in this category is the cloud ontology, which was derived in the frame of the mOSAIC project [25] and aims at providing a transparent and simple access to heterogeneous cloud resources and avoid locked-in proprietary solutions. Other examples also include [5] and [18].
– *Cloud services discovery and selection* – this category consists of ontologies which facilitate the process of discovery and selection of best cloud services. Typically, an ontology serves as a unified common benchmark against which the comparison of various heterogeneous services is performed. Examples of such ontologies include [11,17,19,20,31,38].

Another cluster of related research efforts comprises studies which utilise ontologies to formally describe sensor-enabled domains, collectively referred to as

Semantic Sensor Web (SSW) [28]. Compton et al. [9] provide a survey of 12 SSW ontologies, which provide vocabularies of concepts, relationships between those concepts and built-in reasoning techniques to facilitate semantic interoperability, and compare these ontologies with respect to such criteria as main purpose of use, expressive power, underlying technology, etc.

In this light, a particularly notable and representative example of an ontology used to model sensor networks of any complexity via a common and standardised vocabulary is the Semantic Sensor Network (SSN) ontology, developed by the SSW community. It is a product of careful analysis and comparison of existing sensor ontologies by a group of established researchers and experts in the field [36]. The SSN ontology comprises ten modules, and includes 41 concepts and 39 object properties, which describe sensors, the accuracy and capabilities of such sensors, observations and methods used for sensing, as well as other related concepts [10]. Despite this coverage, the ontology remains domain-independent, as it does not describe domain concepts – these are intended to be defined separately, and included from other linked resources. Such domain independence allows for potential applications of the ontology to a wide range of sensor-enabled domains (for example, the emerging area Internet of Things [29]), and is exploited in the work described in this paper.

Nevertheless, none of the existing cloud ontologies features the sensor-related dimension, and none of the existing sensor ontologies captures the "logical" sensors of CAPS (albeit they offer ways of extending them with relevant concepts). Given this situation, we have developed our own Cloud Sensor Ontology[5] (CSO), which combines the two aforementioned domains, and in the next section we explain how it can be used to express software sensors within CAPs.

4 Cloud Sensor Ontology (CSO)

The principles underpinning the development of the CSO reflected existing insights, best practices, and recommendations as to how sensor-enabled domains should be modeled using ontologies (apart from the SSN ontology, which was the main point of reference in our work, other important influences were OntoSensor [16] and Ontonym [34]). Moreover, when developing the CSO, we tried to follow established ontology engineering principles [26,30], such as clarity, coherence, consistency, extensibility and adoption of naming conventions.

Having outlined some of the key structures defined within CSO, we demonstrate by example how the resulting ontology can be used to define RDF streams, C-SPARQL queries and SWRL rules, thereby helping to detect excessive numbers of client connections to Heroku's Postgres database add-on service.

4.1 Structure of the CSO

When shifting focus from the conventional physical sensor devices of the Sensor Web domain to the "logical software sensors" of CAPs, many of the concepts

[5] Available at `http://seerc.org/ikm/docs/cso.owl`.

defined in existing sensor ontologies become irrelevant and may be omitted. Mainly, these are the concepts related to the physical placement and environment of sensor devices. Additionally, since existing ontologies primarily target sensor observations, they do not include concepts related to situation assessment and adaptations, and this was another challenge for us when developing the CSO.

Logically, CSO can be divided into an upper (i.e., platform-independent) and a lower (i.e., platform-specific) level. The former contains high-level concepts which are potentially reusable across multiple CAPs, whereas the latter contains domain-specific knowledge, such as actual cloud service names and their properties. Accordingly, as far as the principle of *ontology completeness* is concerned, our work on these levels is still ongoing: we are investigating various case studies (one of which will be demonstrated below) with a view to extending and optimising both the upper and the lower parts of the ontology, e.g., to capture concepts relevant to a specific CAP and its services.

The upper ontology includes 5 modules:

– `Sensor` (Fig. 2) – this is the main class used to describe sensors within CAPs, and includes such subclasses as `Service`, `PlatformComponent`, `Application`, `User`, etc.
– `Property` (Fig. 3) – this class describes various qualities of software sensors to be observed, such as `Size` (further sub-classed into `DatabaseSize`, `QueueSize`, etc.); `Time` (further sub-classed into `ExecutionTime`, `QueuingTime`, `StartingTime`, `FinishTime`, etc.); and `NumberOfConnections`. The `Property` class is related to `Sensor` through the `hasProperty` object property, which is further sub-classed into `hasTime`, `hasSize`, `hasNumberOfConnections`, etc. In adopting this structure we have followed the *Sensor-Observes-Property* pattern adopted by the SSN, OntoSensor and Ontonym ontologies. This pattern facilitates conciseness and

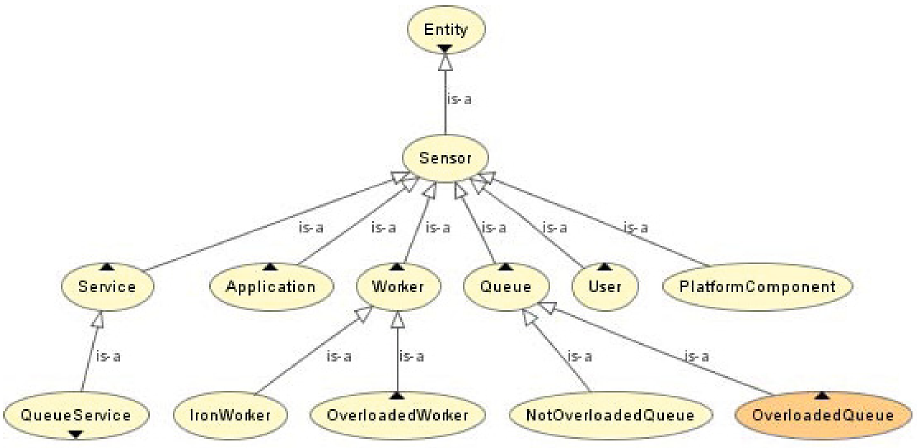

Fig. 2. Upper ontology: the `Sensor` module

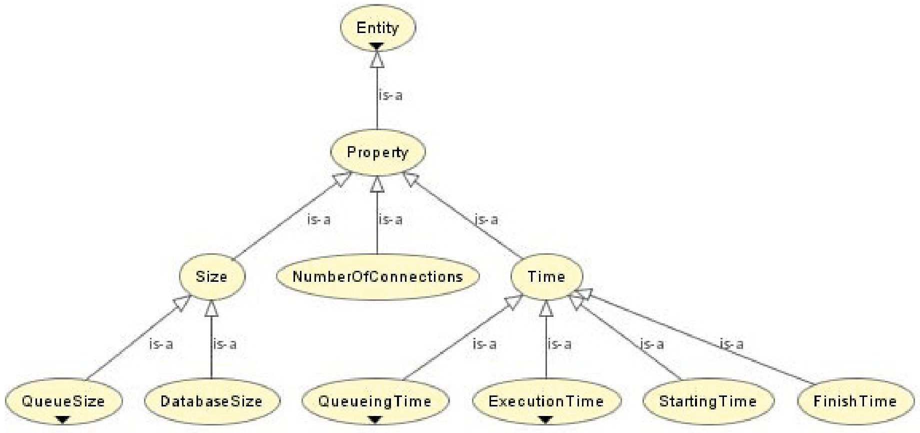

Fig. 3. Upper ontology: the `Property` module

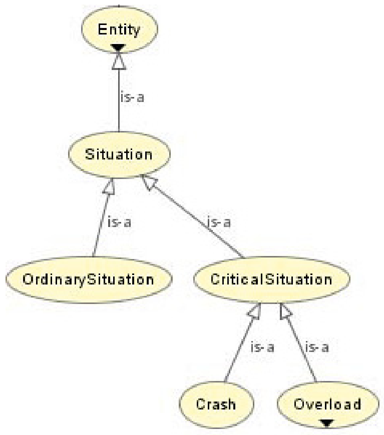

Fig. 4. Upper ontology: the `Situation` module

enables defining the upper concepts (i.e., `Sensor`, `hasProperty`, `Property`) first, and then extending them with required subclasses and sub-properties, thus avoiding redundancy and repetitions.

- `Situation` (Fig. 4) – this class contains the subclasses `CriticalSituation` and `OrdinarySituation`, which are used to classify observations as either requiring or not requiring adaptation actions. `CriticalSituation` includes such subclasses as `Crash`, `Overload`, and `ClientConnectionViolation`.
- `Adaptation` (Fig. 5) – this class defines possible adaptation actions in response to detected critical situations, and includes such subclasses as `ResourceProvisioning`, `ResourceDeprovisioning`, and `Substitution`.
- `Object` (Fig. 6) – this is an auxiliary class to model all other entities within CAPs which should not necessarily be modeled as `Sensors`.

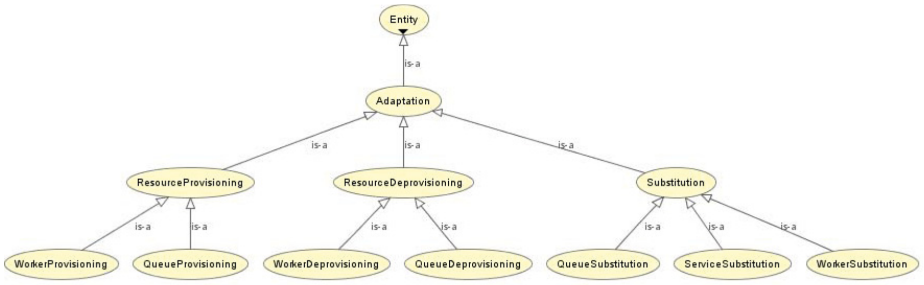

Fig. 5. Upper ontology: the `Adaptation` module

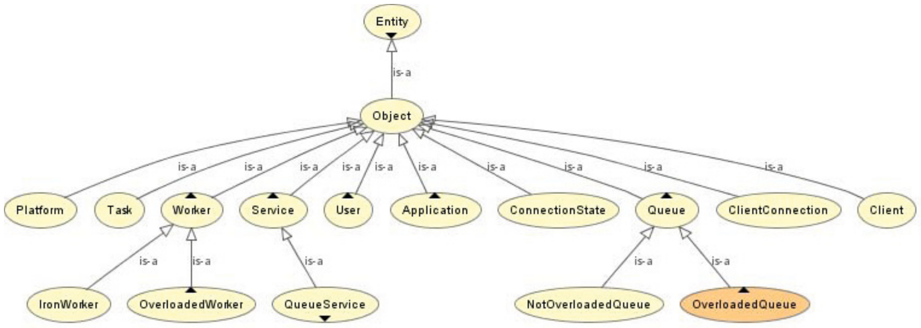

Fig. 6. Upper ontology: the `Object` module

4.2 Example: The Role of the CSO in a Sample Adaptation Loop

We now illustrate how the CSO can be used with RDF sensor streams, C-SPARQL queries and SWRL policies to address existing potential shortcomings of Heroku and its add-on services. For this example, we focus on the Postgres database service[6], one of several data storage services offered by Heroku.

Heroku's pricing model offers customers a range of subscription plans, each offering a different level of service. In particular, a typical metric relating to data storage services is the *number of simultaneous client connections.* However, customers are not currently notified in advance when the number of active connections is reaching 'danger levels', and this can result in further connection requests being unexpectedly rejected. Accordingly, our goal in this case study was to equip data storage services with sensing capabilities, so that application providers can be notified in advance whenever a threshold is approaching, allowing them to take appropriate preemptive actions – for example, by closing down low-priority connections or by automatically upgrading their subscription plan.

Using our framework we manually annotated sensor data (in this case, the current pool of client connections and the current state of the database backup

[6] `https://www.heroku.com/postgres`

process) with semantic descriptions defined in the CSO to generate a homogeneous data representation, and then streamed these RDF values to the C-SPARQL querying engine.[7] The following RDF stream captures the situation when the number of client connections increased from 15 to 18 (the connection limit is 20 for the initial subscription plan), and no backup process is running – this is important because the backup process establishes two client connections to the database, but typically lasts for less than a minute, and therefore should not be considered as a threat.

```
cso:postgres-service-10 rdf:type cso:StorageService
cso:postgres-service-10 cso:hasNumberOfConnections
    cso:number-of-connections-122
cso:number-of-connections-122 rdf:type cso:NumberOfConnections
cso:number-of-connections-122 cso:hasValue "15"^^xsd:int

cso:postgres-service-10 rdf:type cso:StorageService
cso:postgres-service-10 cso:hasNumberOfConnections
    cso:number-of-connections-122
cso:number-of-connections-122 rdf:type cso:NumberOfConnections
cso:number-of-connections-122 cso:hasValue "16"^^xsd:int

cso:backup-service-8 rdf:type cso:BackupService
cso:backup-service-8 cso:accesses cso:postgres-service-10
cso:backup-service-8 rdf:isActive "false"^^xsd:boolean

cso:postgres-service-10 rdf:type cso:StorageService
cso:postgres-service-10 cso:hasNumberOfConnections
    cso:number-of-connections-122
cso:number-of-connections-122 rdf:type cso:NumberOfConnections
cso:number-of-connections-122 cso:hasValue "18"^^xsd:int
```

In order to assess current situation and detect violations we registered a standing C-SPARQL query, which is evaluated every second and triggered whenever the number of client connections during the previous minute reaches the threshold of 18, provided there is no backup process running – that is, there are indeed 18 client connections, and there is a potential threat to the application stability.

```
REGISTER QUERY PostgresClientConnectionViolation
AS PREFIX cso:<http://seerc.org/ontology.owl#>
SELECT ?service1, ?noc
FROM STREAM <http://seerc.org/stream> [RANGE 1m STEP 1s]
WHERE { ?service1 rdf:type cso:HerokuPostgresService .
    ?service1 cso:hasNumberOfConnections ?noc .
    ?noc cso:hasValue ?v . FILTER (?v >= 18) .
```

[7] To extract these metrics from the Postgres service we relied on standard mechanisms offered by this database. See [13].

```
    ?service2 rdf:type cso:PGBackupService .
    ?service2 cso:accesses ?service1.
    ?service2 cso:isActive "false"^^xsd:boolean }
```

Once the C-SPARQL query is triggered, the corresponding critical values are instantiated in the CSO as instances of `HerokuPostgresService` and `CriticalNumberOfConnections`, and reasoning over SWRL rules is applied. The following two rules define that: (a) a situation when a Postgres service has a critical number of client connections should be classified as critical under the `ClientConnectionViolation` class; and (b) such a critical situation requires an adaptation – in this case a subscription plan upgrade.

```
HerokuPostgresService(?ser), CriticalNumberOfConnections(?noc),
    hasNumberOfConnections(?ser, ?noc), Situation(?sit)
          -> ClientConnectionViolation(?sit)

HerokuPostgresService(?ser), ClientConnectionViolation(?sit),
    isInSituation(?ser, ?sit)
          -> needsSubscriptionPlanUpgrade(?ser, true)
```

5 Next Steps: Linked Data

Our original motivation to employ the Semantic Web technology stack was to move away from rigid, hard-coded approaches and introduce a flexible, easily maintainable, and platform-independent way of expressing diagnosis and adaptation policies for the domain of CAPs. Our self-adaptation framework offers CAP providers an opportunity to define policies in a declarative and human-readable manner by using the underlying Cloud Sensor Ontology as a common vocabulary of terms.

However, as illustrated in the Postgres example above, this approach implies that cloud providers are responsible for maintaining adaptation-related knowledge which concerns not only the internal platform components, but also third-party services, which are registered with the given CAP. In reality, however, this is not necessarily the case. Typically, third-party service providers, having deployed their software on a cloud and exposing the API to the users, take on the responsibility to maintain the software and associated resources, and provide customers with required support. This means that CAP providers treat third-party services as black boxes and need not be aware of their internal architecture and organisation. Accordingly, this may result in situations where adaptation policies are incomplete, imprecise, or even invalid, which in turn may lead to incorrect adaptations, non-optimised resource consumption, and even system failures.

As a potential solution to this problem we are currently investigating how *Linked Data* principles can be utilised in this context. The primary goal of Linked Data is to enable discovery and sharing of semantically-enriched data over the Web using standardised technologies, such as URIs and RDF [6]. In other words, Linked Data implies the ubiquitous re-use of existing distributed data, which is exactly what we need in order to separate various pieces of adaptation policies between CAP owners and third-party service providers.

Accordingly, we believe that using Linked Data principles will enable us to create a distributed two-level ontological framework, which would consist of:

– *Platform Adaptation Ontology*: a core OWL ontology containing all the necessary concepts, relations and default SWRL rules needed to define the default adaptation-related behaviours of platform components and services.
– *Extension Adaptation Ontologies*: a set of linked OWL ontologies and SWRL rules developed by third-party service providers and deployed on the Web, which specify diagnosis and adaptation policies for respective services registered with a CAP. These ontologies may either extend or overwrite the default behaviour specified in the core Platform Adaptation Ontology.[8]

The main benefits of Linked (Open) Data are that it is *sharable*, *extensible*, and easily *re-usable*. In the context of a distributed ontological framework for adaptation policies, we also postulate the following additional benefits:

– Linked extensions are distributed and easily accessible over the Web by means of URIs and/or SPARQL endpoints. In this sense, software services become "self-contained" as they inform the autonomic manager about their diagnosis/adaptation policies by providing a link to the corresponding policies. The autonomic management system need not know about them in advance, but can access them at run-time using Linked Data principles.
– Linked extensions are easily modifiable. Since third-party service providers have full control over their segment of policies, they can seamlessly change them so as to reflect ongoing changes.
– Linked extensions are potentially re-usable across multiple CAPs. Indeed, it is quite common for third-party service providers to offer their services on several CAPs. For example, CloudAMQP[9] – a messaging queue service – is offered on 10 different CAPs (including Amazon Web Services[10], Heroku, Google Cloud Platform[11], etc.). Accordingly, under certain assumptions one and the same policy definition can be re-used across all of those CAPs.

Moreover, there is no need to restrict oneself to exposing as Linked Data only the schemas. In the future we may consider publishing historical CAP sensor observations (as homogenised RDF triples) in public repositories. Indeed, we already record these datasets for the purposes of post-mortem analysis, in

[8] It should be perhaps noted that typically the term "Linked Data" refers to published RDF datasets (or "instance data"), rather than to OWL, RDFS and SWRL vocabularies. Nevertheless, there is ongoing research aimed specifically at linking, sharing and re-using the underlying schemas, not just the datasets themselves. See, for example, Linked Open Vocabularies - LOV (`http://lov.okfn.org/dataset/lov/`) for a representative example of this research initiative.

[9] `http://www.cloudamqp.com/`

[10] `https://aws.amazon.com/`

[11] `https://cloud.google.com/`

order to identify critical failures or suboptimal behaviours. We are also planning to apply machine learning techniques so as to detect underlying trends and patterns, which will hopefully lead to more precise and accurate diagnosis, and more efficient adaptation policies. Exposing this information as Linked Data will, we believe, provide access to real-world performance measurements, and will have the potential to facilitate comparison between different CAPs.

6 Summary and Conclusions

In this paper we have presented our Cloud Sensor Ontology (CSO), and illustrated how it lies at the core of our semantic self-adaptation framework for cloud application platforms. Having Despite thoroughly studied studying related ontologies in the domains of SSW and cloud computing, we were unable to find one, which would satisfy our the requirements of sensor-enabled cloud application platforms "out of the box", and therefore opted to define our own. Taking the established SSN ontology as a reference, we developed this ontology by treating CAPs as networks of distributed "logical" software sensors. We employed the CSO as an underlying semantic architecture model of cloud environments as well as a shared vocabulary of terms for defining RDF sensor streams, C-SPARQL continuous queries for performing situation assessment, and SWRL rules for the final diagnosing and adaptation planning. We have argued that this approach allows us to benefit from an extensible architecture to introduce new software services, flexible and declarative declaration of adaptation policies, and powerful reasoning capabilities to analyse critical situations and suggest possible adaptation strategies. Accordingly, we believe that the CSO might be of interest to academic or industrial researchers willing to ontologically model complex software environments as networks of distributed logical sensors.

On the other hand, we can also identify potential shortcomings of the approach. These include the general scalability issue of formal reasoning, which in presence of large amounts of monitored data within CAPs needs to be properly addressed [23]. We are planning to perform a more formal evaluation of the CSO with respect to its scalability and "queriability" – that is, how the shape of the ontology affects the performance, and whether potential downgrades can be tolerated in favour of increased analytical and reasoning capabilities of the described approach.

We are also planning to experiment with another CAP to prove flexibility and agility of our approach. In these circumstances, another issue to be considered is the proprietary software standards, which may restrict us from inserting probes into applications to extract monitoring data.

To address the potential shortcoming associated with CAP providers being not necessarily in a position to define policies concerning a particular third-party service registered with the given CAP, we also proposed utilising Linked Data principles so as to decouple semantic knowledge concerning the CAP per se from knowledge concerning external services. This approach has the potential to create an open extensible architecture where a cloud sensor network consists of independent self-contained sensors (i.e., platform components and services), described by a two-tier distributed set of interacting ontologies.

References

1. Androcec, D., Vrcek, N., Seva, J.: Cloud Computing Ontologies: A Systematic Review. In: MOPAS 2012, The Third International Conference on Models and Ontology-based Design of Protocols, Architectures and Services, pp. 9–14 (2012)
2. Barbieri, D.F., Braga, D., Ceri, S., Della Valle, E., Grossniklaus, M.: C-SPARQL: SPARQL for continuous querying. In: Quemada, J., León, G., Maarek, Y.S., Nejdl, W. (eds.) Proceedings of the 18th International Conference on World Wide Web, WWW 2009, Madrid, Spain, April 20-24, pp. 1061–1062. ACM, New York (2009)
3. Berners-Lee, T.: Semantic Web on XML (2000), `http://www.w3.org/2000/Talks/1206-xml2k-tbl`
4. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American 284(5), 34–43 (2001)
5. Bernstein, D., Vij, D.: Intercloud Directory and Exchange Protocol Detail Using XMPP and RDF. In: 2010 6th World Congress on Services (SERVICES-1), pp. 431–438 (July 2010)
6. Bizer, C., Heath, T., Berners-Lee, T.: Linked Data – The Story So Far. International Journal on Semantic Web and Information Systems (IJSWIS) 5(3), 1–22 (2009)
7. Böhm, M., Leimeister, S., Riedl, C., Krcmar, H.: Cloud computing – outsourcing 2.0 or a new business model for it provisioning? In: Keuper, F., Oecking, C., Degenhardt, A. (eds.) Application Management, pp. 31–56. Gabler (2011)
8. Botts, M., Percivall, G., Reed, C., Davidson, J.: OGC® Sensor Web Enablement: Overview and High Level Architecture. In: Nittel, S., Labrinidis, A., Stefanidis, A. (eds.) GSN 2006. LNCS, vol. 4540, pp. 175–190. Springer, Heidelberg (2008)
9. Compton, M., Henson, C.A., Neuhaus, H., Lefort, L., Sheth, A.P.: A Survey of the Semantic Specification of Sensors. In: Proc. Semantic Sensor Networks 2009, pp. 17–32 (2009)
10. Compton, M., Barnaghi, P., Bermudez, L., Garcia-Castro, R., Corcho, O., Cox, S., Graybeal, J., Hauswirth, M., Henson, C., Herzog, A., Huang, V., Janowicz, K., Kelsey, W.D., Phuoc, D.L., Lefort, L., Leggieri, M., Neuhaus, H., Nikolov, A., Page, K., Passant, A., Sheth, A., Taylor, K.: The SSN Ontology of the W3C Semantic Sensor Network Incubator Group. Web Semantics: Science, Services and Agents on the World Wide Web 17 (2012), `http://www.websemanticsjournal.org/index.php/ps/article/view/312`
11. Dastjerdi, A.V., Tabatabaei, S.G.H., Buyya, R.: An Effective Architecture for Automated Appliance Management System Applying Ontology-Based Cloud Discovery. In: 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid 2010), pp. 104–112 (May 2010)
12. Dautov, R., Paraskakis, I.: A vision for monitoring cloud application platforms as sensor networks. In: Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference, pp. 25:1—25:8. ACM, Miami (2013)
13. Dautov, R., Paraskakis, I., Stannett, M.: Towards a Framework for Monitoring Cloud Application Platforms as Sensor Networks. Cluster Computing Journal (in press, 2014)

14. Dautov, R., Kourtesis, D., Paraskakis, I., Stannett, M.: Addressing Self-management in Cloud Platforms: A Semantic Sensor Web Approach. In: Proceedings of the 2013 International Workshop on Hot Topics in Cloud Services, HotTopiCS 2013, pp. 11–18. ACM, New York (2013)
15. Fortis, T.F., Munteanu, V.I., Negru, V.: Towards an ontology for cloud services. In: 2012 Sixth International Conference on Complex, Intelligent and Software Intensive Systems (CISIS), pp. 787–792 (July 2012)
16. Gruber, T.R.: Toward Principles for the Design of Ontologies Used for Knowledge Sharing. Int. J. Hum.-Comput. Stud. 43(5-6), 907–928 (1995)
17. Han, T., Sim, K.M.: An ontology-enhanced cloud service discovery system. In: Ao, S.I., Castillo, O., Douglas, C., Feng, D.D., Lee, J.A. (eds.) Proceedings of the International MultiConference of Engineers and Computer Scientists 2010, IMECS, Hong Kong, March 17-19, vol. I, pp. 644–649. Newswood Limited/International Association of Engineers (2010)
18. He, K.Q., Wang, J., Liang, P.: Semantic Interoperability Aggregation in Service Requirements Refinement. Journal of Computer Science and Technology 25(6), 1103–1117 (2010)
19. Kang, J., Sim, K.M.: Towards Agents and Ontology for Cloud Service Discovery. In: 2011 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), pp. 483–490 (October 2011)
20. Kang, J., Sim, K.M.: Cloudle: An Ontology-Enhanced Cloud Service Search Engine. In: Chiu, D.K.W., Bellatreche, L., Sasaki, H., Leung, H.-F., Cheung, S.-C., Hu, H., Shao, J. (eds.) WISE Workshops 2010. LNCS, vol. 6724, pp. 416–427. Springer, Heidelberg (2011)
21. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. Computer 36(1), 41–50 (2003)
22. Kourtesis, D., Bratanis, K., Bibikas, D., Paraskakis, I.: Software Co-development in the Era of Cloud Application Platforms and Ecosystems: The Case of CAST. In: Camarinha-Matos, L.M., Xu, L., Afsarmanesh, H. (eds.) Collaborative Networks in the Internet of Services. IFIP AICT, vol. 380, pp. 196–204. Springer, Heidelberg (2012)
23. Le-Phuoc, D., Dao-Tran, M., Xavier Parreira, J., Hauswirth, M.: A native and adaptive approach for unified processing of linked streams and linked data. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 370–388. Springer, Heidelberg (2011)
24. Ma, Y.B., Jang, S.H., Lee, J.S.: Ontology-based resource management for cloud computing. In: Nguyen, N.T., Kim, C.-G., Janiak, A. (eds.) ACIIDS 2011, Part II. LNCS, vol. 6592, pp. 343–352. Springer, Heidelberg (2011)
25. Moscato, F., Aversa, R., Di Martino, B., Fortis, T., Munteanu, V.: An analysis of mOSAIC ontology for Cloud resources annotation. In: 2011 Federated Conference on Computer Science and Information Systems (FedCSIS), pp. 973–980 (September 2011)
26. Russomanno, D.J., Kothari, C.R., Thomas, O.A.: Building a Sensor Ontology: A Practical Approach Leveraging ISO and OGC Models. In: The 2005 International Conference on Artificial Intelligence, Las Vegas, NV, USA, pp. 637–643 (2005)
27. Rymer, J.R., Ried, S., Matzke, P., Magarie, A., Anderson, A., Lisserman, M.: The Forrester Wave^{TM}: Platform-As-A-Service For Vendor Strategy Professionals, Q2 2011 – A BT Futures Report: Identifying The Best Partner Choices For ISVs And Service Providers. Business report, Forrester Research (May 19, 2011)

28. Sheth, A., Henson, C., Sahoo, S.S.: Semantic Sensor Web. IEEE Internet Computing 12(4), 78–83 (2008)
29. Soldatos, J., Serrano, M., Hauswirth, M.: Convergence of Utility Computing with the Internet-of-Things. In: Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS 2012), pp. 874–879 (2012)
30. Stevenson, G., Knox, S., Dobson, S., Nixon, P.: Ontonym: A Collection of Upper Ontologies for Developing Pervasive Systems. In: Proceedings of the 1st Workshop on Context, Information and Ontologies, CIAO 2009, pp. 9:1–9:8. ACM, New York (2009)
31. Tahamtan, A., Beheshti, S., Anjomshoaa, A., Tjoa, A.: A Cloud Repository and Discovery Framework Based on a Unified Business and Cloud Service Ontology. In: IEEE Eighth World Congress on Services (SERVICES 2012), pp. 203–210 (2012)
32. Takahashi, T., Kadobayashi, Y., Fujiwara, H.: Ontological approach toward cybersecurity in cloud computing. In: Proceedings of the 3rd International Conference on Security of Information and Networks, SIN 2010, pp. 100–109. ACM, New York (2010)
33. Tsai, W.T., Sun, X., Balasooriya, J.: Service-Oriented Cloud Computing Architecture. In: Proceedings of the 2010 Seventh International Conference on Information Technology: New Generations, ITNG 2010, pp. 684–689. IEEE Computer Society, Washington, DC (2010)
34. Uschold, M., Gruninger, M.: Ontologies: Principles, methods and applications. Knowl. Eng. Rev. 11, 93–136 (1996)
35. Uschold, M., Gruninger, M.: Ontologies and semantics for seamless connectivity. SIGMOD Rec. 33(4), 58–64 (2004)
36. W3C Semantic Sensor Network Incubator Group: Review of sensor and observations ontologies (June 17, 2011)
37. Youseff, L., Butrico, M., Da Silva, D.: Toward a Unified Ontology of Cloud Computing. In: Grid Computing Environments Workshop, GCE 2008, pp. 1–10 (November 2008)
38. Zhang, M., Ranjan, R., Nepal, S., Menzel, M., Haller, A.: A Declarative Recommender System for Cloud Infrastructure Services Selection. In: Vanmechelen, K., Altmann, J., Rana, O.F. (eds.) GECON 2012. LNCS, vol. 7714, pp. 102–113. Springer, Heidelberg (2012)