Semantics-based Plug-and-Play Configuration of Sensor Network Services

Mukaddim Pathan, Kerry Taylor and Michael Compton

CSIRO ICT Centre, Canberra, ACT 2601, Australia
{mukaddim.pathan, kerry.taylor, michael.compton}@csiro.au

Abstract. A sensor network is expected to provide effective delivery of its services by taking an appropriate action based on one or more situations that it senses in the environment. However, due to the dynamism of application requirements and user context, it is often required to re-configure services from a sensor network to meet specific application needs. This paper is an attempt to enable dynamic adaptation of sensor network services to environmental changes with minimal human intervention. We propose a semantics-based architecture that uses a publish/subscribe pattern to enable self-configuration of sensor network services and potentially alleviate the heavy burden of programming sensor networks. We envisage an environment with the plug-and-play concept, where sensor networks, sensor nodes and sensors are dynamically reconfigured, ensuring minimal disruption to the delivery of higher order information services. We thus allow a client application to interact with the system and uniformly access sensor services, yet hiding the complexity of the system.

1. Introduction

Recent years have witnessed the emergence of computationally-enabled sensors, along with the rapid development and deployment of applications that use sensor data [1, 39]. A sensor network perceives the environment, monitors different parameters and gathers data according to an application purpose. The capabilities of a sensor network are not just limited to observing and forwarding raw sensor readings. Sensor networks revolutionize sensing in a wide range of application domains. They provide different services, ranging from home automation to process monitoring, healthcare analysis, weather forecasting, military situation awareness, and traffic control.

Sensor networks are often used in uncontrollable environments, whereby users may not have precise information about the sensing data or network characteristics [2]. Moreover, these properties can change over the lifetime of a network deployment, due to events such as signal quality degradation, isolated failures, resource addition, and sensor movement. As a consequence, sensor services must be re-configured to make use of the new context information and resources. There should be the provision to autonomously adapt, i.e. *self-configure*, the sensor network services according to application scenarios and context, with enhanced accessibility and reusability.

The self-configuration feature in a sensor network would enable autonomous structuring of contextual information and resources thus making them available to

services. It is proposed as one of the means to make a system scalable and robust in the presence of changes, supporting dynamic adaptation [17]. It also allows service customization and supports employing semantic technologies. A self-configurable sensor network service can be built through a combination of process changes, technology evolution, architecture and open industry standards.

Although there are several research works in the autonomic computing [12, 22], model-driven engineering [31, 38], overlay networks [21], service computing [20, 36], and sensor network [2, 5, 7] domains to deal with the self-* properties of a system, there is the need for a detailed architecture for self-configuration of sensor services. Our research is a step towards addressing this issue by enabling a sensor network to understand the general and specific context it operates in, i.e. the physical, environmental and application-specific context. The underlying focus is to orchestrate system activities according to an individual context, so as to provide a range of self-configurable services that are not possible with the current sensor configurations.

We propose a rich and powerful semantics-based architecture to recognize relevant environmental changes and thus assist the self-configuration of sensor network services. The need for a bottom-up sensor network description is addressed by taking advantage of the W3C SSN-XG ontology[1]. This OWL [11] ontology defines sensing concepts as axioms in the TBox and grounded data about particular configurations as ABox instances. In addition, a semantic-based publish/subscribe approach is followed for self-registration of configuration details such that high level network services, e.g. databases, reporting services and user-specified tasks, may be informed of changes of significance to them so they can adapt their behavior accordingly. For example, with the aid of our approach, the introduction of an additional thermistor to a sensor node measuring soil temperature may cause an application program to be recompiled and re-deployed on the sensor network to use the additional capability. The main contributions of this paper are:

1. An architectural description for self-configuration of sensor network services in order to alleviate the heavy burden of programming sensor networks.
2. A semantic-based publish/subscribe approach as a building block to support dynamic service adaptation to changes.
3. An approach for client applications to interact with a wide range of services in a common way, thus providing a transparent view of the system complexity.

The rest of this paper is structured as follows. Section 2 sets the stage by detailing design considerations and our approach for self-configuration of sensor services. It is followed by a comparative analysis of related work to highlight the novelty of our work. The proposed architecture and associated research challenges are presented in Section 4 and Section 5, respectively. The system model is detailed in Section 6 and implementation plan follows next. The paper is concluded in Section 8.

2. Self-Configuration of Sensor Network Services

The self-configuration ability can assist a sensor network service to dynamically adapt to the changing environment, including the deployment of new components or the

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

removal of existing ones, or sudden changes in the system characteristics. These variations can happen either as random, periodic events or gradual evolutionary changes in the system. The underlying principle for building self-configuring sensor network services is to provide fundamental mechanisms upon which other networking and system services may be spontaneously specified and reconfigured [20]. To make a sensor network service self-configurable, the following conditions must be met:

- The tasks involved in the configuration of a service are automated.
- The automation process is initiated based on situations that can be observed or detected in the sensor network system.
- An authoritative entity in the sensor network must possess sufficient knowledge of sensor network service configuration, resources, policies, and observed data.

On satisfying these conditions, it is possible to assemble the automated functions in a set of composed processes to allow a sensor network service to be self-configurable. These processes can be governed to collect necessary details from the system, analyze them to determine the required configuration changes for a sensor service, create a plan or action sequence specifying the necessary changes, and perform those actions.

2.1. Our Approach

Semantic technologies [28] are often used to enable sensor network services and applications by providing a universally accessible platform that allows data to be shared and processed by automated tools, and by providing machine-understandable semantics of sensed data for automatic information processing and exchange. The use of semantic technologies assists the integration of heterogeneous sensor resources that need to interoperate, and support the automation of service configurations. This semantic-based approach can provide a powerful way to express the context and sensor capabilities, thus serving the vision of self-configuration of sensor network services. It is particularly applicable as the full service description of a sensor network system can never be known in advance.

To enable self-configuration of a sensor service, it is required to: i) define the service configuration in a standard, machine readable format; ii) publish the definition via a registry in order to make it discoverable; and iii) support a protocol to communicate with client applications. We follow a semantic-based publish/subscribe approach to list the available configurations. A repository provides necessary information about the underlying sensor networks in terms of observations, measurements, and sensor capabilities to be used for facilitating the self-configuration activities when certain events occur. The publish/subscribe interactions well adapt to the needs of self-configuration of sensor services. The main goal of such a mechanism is the decoupling of publishers and subscribers in time, space and synchronization.

An ontology [8] can be used to describe various aspects of a sensor network service and bridge the gap between services (re)usable in various scenarios and requirements in the context of an individual client application. An ontology provides a medium for capturing and reusing the knowledge and experience gained from prior efforts, it thus leads to a great level of automation at the semantic level. Our approach may also involve using “smart-sensors” capable of self-describing and directly

providing intermediate features and capabilities using ontology so as to seamlessly enable activity recognition when placed into an existing infrastructure.

Dynamic adaptation copes with changes in sensor configurations and application context. When there are changes, the runtime information is fed into the related self-configuration context ontology using a feedback mechanism. The changes trigger the execution of self-configuration logic for dynamic adaptation via monitoring, analysis, planning and execution. With this approach, we aim to achieve extensibility, performance, and scalability and lay a foundation for the self-configuration activities.

3. Related Work

Several research works focus on semantic-based dynamic service adaptation, sensor configuration programming and the creation of context-oriented overlays. In this section, we first cover the programming approaches in existing literature, and then provide a comparative analysis of related research work.

3.1. Programming Models

Existing literature [23, 32] reports several approaches for programming a sensor network. In *full-image code update* [30], a monolithic binary image of an entire application is generated to fit into a single packet and is distributed by the source to all nodes within a broadcast domain. *Partial-image code update* [13] calculates code differences to be integrated into the program image and deliver them to target nodes. It provides similar flexibility as full-image update, however, at a significantly lower cost. In a *node-level* approach, programs are generally assembled on a server by combining application specific code, library routines, configuration information, and operation system into a single image that can be delivered to a node. Some middleware performs *dynamic update* by replacing or reconfiguring components with the use of Virtual Machines (VMs). VMs provide a more cost efficient approach to update application level functionality of the system. However, the VM scripts are highly restricted in the flexibility of updates [2]. The *Marcoprogramming* approach involves writing a single program for the entire network, rather than for individual nodes in the network. The program is then de-globalized by the compiler to convert it into elements for distributing to individual nodes [19, 32]. There also exists work using semantic representations based on an ontology to support the programming of a heterogeneous sensor network [33].

In our work, we endeavor to use declarative macroprogramming approach for an overlay architecture to enable self-configuration of sensor network services. This is due to the flexibility and convenience of programming overlays in a highly compact and reusable form [21]. The programmer specifies the logic of the program, typically in some variant of first order logic. Then it is left up to the execution engine to supply a control flow that implements the logic. Specifically, we aim to use Snlog [6], a deductive declarative programming language, which is derived from declarative models for building overlays. We believe that our approach promises the advantages of ease of specification and code reuse.

3.2. Overview of Related Systems

Most existing methodologies for system and service configurations are focused on a specific engineering technique. Some of them are largely not extensible and do not support dynamic adaptation to the specific characteristics of individual application scenarios and context.

Semantically-enabled Heterogeneous Service Architecture and Platforms Engineering (SHAPE) [31] provides an integrated development environment that brings together the Model-Driven Engineering (MDE) methodology with the Service-Oriented Architecture (SOA) paradigm. The Real World Internet [25] project focuses on the management, scalability, and heterogeneity of devices and users. It aims to facilitate the dynamic creation of context and actuation services from elementary sensing, actuation, processing and reuse of sensor resources for a large number of applications. The OPPORTUNITY [27] project aims to build goal-oriented sensor assemblies that are spontaneously arising and self-organizing to achieve a common activity or context recognition goal. Hydra [41] follows a semantic Web-based self-management approach, supported by a set of self-management context ontologies. The CONNECT [29] project focuses on enabling automated protocol mediation through a formalization technique to enable seamless system composition. The SANY consortium [14] provides a standard open architecture and a set of basic services for integration of sensors, sensor networks, and sensor services. It also recognizes the OGC's Sensor Web Enablement (SWE) [3] as one of the key technologies to support self-organizing and self-healing in sensor networks.

While our work is related to these research projects, we differ in that the service customization and configuration techniques in the above projects are limited to low-level technical aspects, whereas we seek to insulate client applications from the low-level details. Some of the above works use SensorML or XML-based descriptions of sensor configurations, which is not usable in our context due to not supporting reasoning and vocabulary of semantic descriptions. In addition, we seek to provide a comprehensive mapping of the SSN-XG ontology and OGC SWE standards. Many of the design considerations, i.e. structural and operational requirements, for our architecture are similar to that of ORCHESTRA [36], which provides a service-oriented spatial data infrastructure. Our work also has some similarity to the P2 system [21] that uses a declarative logic language to express overlays in a highly compact and reusable form.

In recent years, a few research work have explored the use of semantics for sensor networks and publish/subscribe systems, such as semantic-based service framework for query processing [18], sensor plug and play [4], semantic filtering in an XML-based publish/subscribe infrastructure [35], an ontology-based publish/subscribe system [37], semantic-enabled publish/subscribe middleware [9, 26], and semantic-based publish/subscribe systems [24, 40]. While they are appealing, none of them focuses on the self-configuration aspect as we do. Many of them do not make use of reasoning or domain knowledge in ontologies to aid the identification of semantically relevant service configurations and matching them to subscribers. Moreover, most of them suffer from scalability limitations with the increase of system size, and usually do not work well under a large number of application subscriptions and a high volume of service configurations. In our work, we aim to address these limitations.

4. The Architecture

The aim of self-configuration is that the service will re-configure itself so as to again either satisfy the changed application context, scenario, and/or environment. The requirement specification for a self-configurable sensor network service is not only functional behavior, but also those non-functional properties such as response time, performance, reliability, security, and that requirement may well include optimization.

Based on the design considerations (Section 2.1), in Figure 1, we present an architecture for self-configuration of sensor services, comprising three key layers:

***Sensor network physical infrastructure*:** At this layer, heterogeneous sensor nodes are assembled and are placed at distributed locations. Each sensor node contains a battery power source, wireless communications, computation unit, multiple sensing modality, and memory. Sensor nodes produce a large amount of environmental data. Collected data from sensor networks are often archived or streamed as raw data, but can be annotated with metadata, by an application-specific data enrichment service from the upper layer. Meaning of sensor data includes the feature of interest, the specification of measuring devices, accuracy, scenario of measurements, and location.

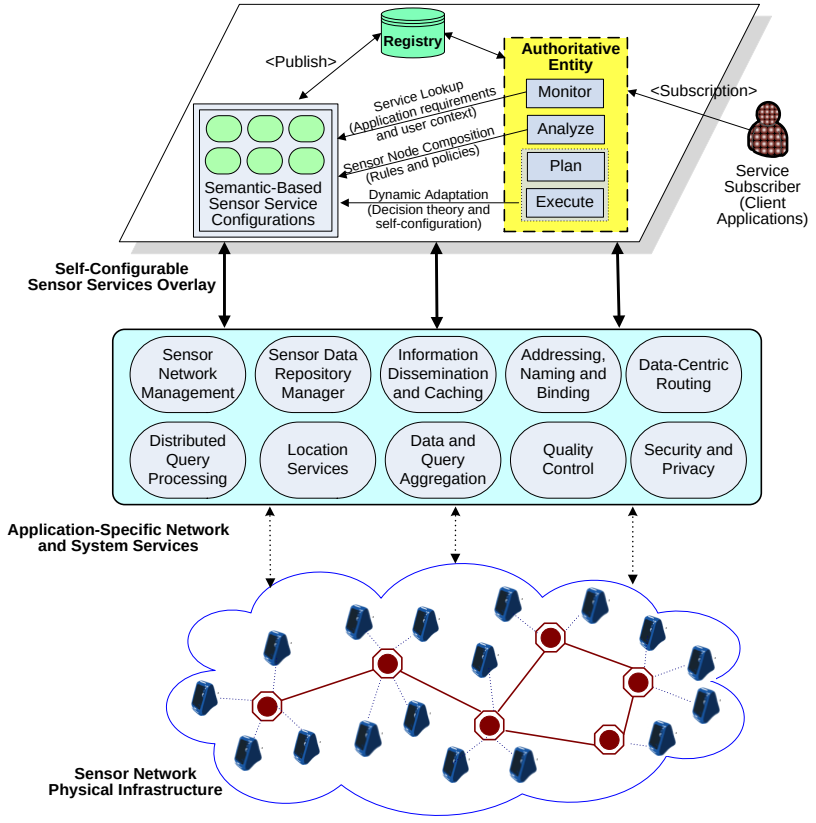


Fig. 1. Architecture to enable self-configuration of sensor network services.

***Application-specific network and system services*:** This layer provides network and systems services to support self-configuration. Among others, they may include data-centric routing, attribute-based addressing mechanisms, location services, naming and binding services, information dissemination and aggregation, caching and security services. These services would use simpler communication interfaces and abstraction than the raw communication interface. They should aid to enhance the overall performance, such as throughput and delay.

***Self-configurable sensor network services overlay*:** This layer forms the basic functionalities of a semantic-based publish/subscribe system. At the heart of this layer are an Authoritative Entity (AE) and a central registry. The registry publishes sensor capabilities and semantic-based service configurations according to an ontology, and allows client applications to subscribe to them. The registry can be user defined in-memory, governance repository or database-driven. AE interacts with this registry and governs the operations of sensor network services. A semantic matching capability [35] is deployed to notify matched subscribers about published service configurations. Upon receiving notification on an observed change, a sensor network service adapts to the modified context, i.e. self-configure itself, by going through monitoring, analyzing, planning and execution processes. These processes are managed by AE. We detail on these processes in the next section.

4.1. Self-Configuration Management

The automated processes provided by AE to manage self-configuration are:

Monitoring process: It provides active monitoring to collect, aggregate, filter and report details gathered from a sensor network system. The data collected by this process may include information about application scenarios, context, service configuration, status, offered capability and throughput. Some of the data is static or changes slowly, whereas other data is dynamic, changing continuously over time.

Analyzing process: It provides the mechanisms to correlate and model complex situations, e.g. time series forecasting and queuing models, to allow AE to learn about the system and help predict future situations. For example: the requirement to enforce a change may occur when the analyzing process determines that there is a modified service configuration to meet the application context.

Planning process: It provides a mechanism to construct the actions needed to satisfy an application context. It takes on many forms ranging from a single command to a complex workflow. It generates the change plan, a set of modifications for services, and logically passes the change plan to the execution process.

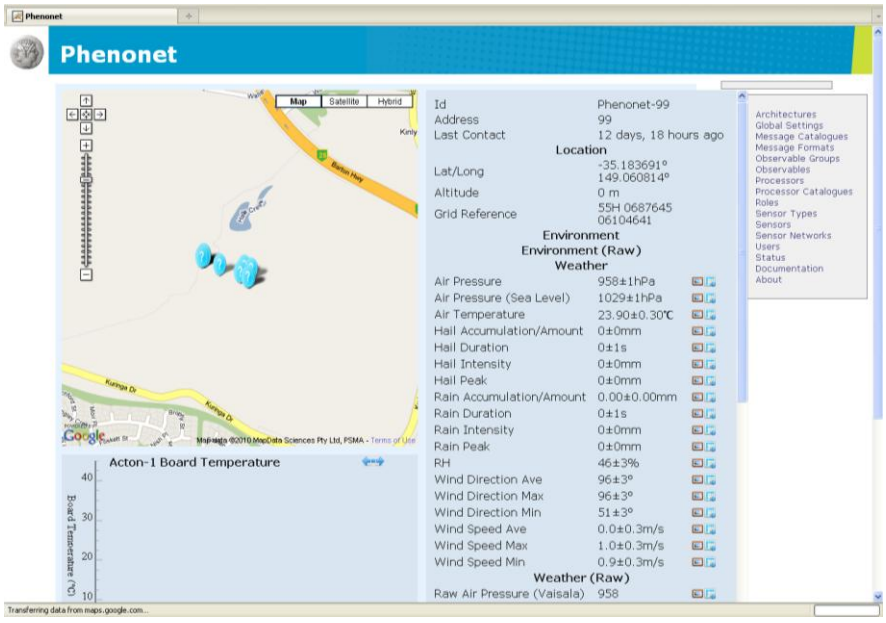
Execution process: It provides the mechanisms to control the execution of a plan with considerations for self-configuration. Once a plan is generated corresponding to a change request, some actions may need to take place to modify the configuration of one or more services. This process is responsible for carrying out the procedure that was generated by the planning process through a series of actions.

The above processes can be tied with three fundamental mechanisms employed within AE. They are—*service lookup* to keep track of the availability of service configurations; *sensor node composition* to provide abstractions of the hierarchical composition of the sensor network, thus simplifying dynamic reconfiguration of

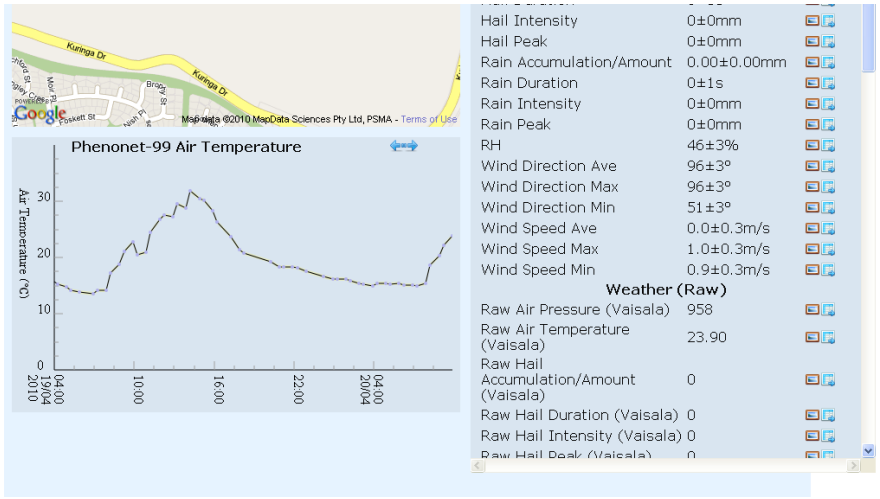
services; and *dynamic adaptation* to utilize information from the lookup and composition services for re-configuration. Through a distributed implementation of these mechanisms sensor services can be enabled to dynamically adapt to changes, e.g. device addition, failure and degradation, movement of sensor nodes, and variation in tasks and network requirements in the sensor network system.

4.2. A Usage Scenario

As a usage scenario of the self-configuration service architecture, we consider the *Phenonet* project [10]. The project deploys a 40-node, distributed sensor network (Figure 2) to monitor crop growth. Amongst other parameters the network observes air and soil temperature, soil moisture, light intensity, solar radiation, and a variety of water quality parameters. The plant scientists involved in this project are interested in comprehensive sensor readings, based on a number of environment and water quality parameters, to run on yield and measure performance after frost, heat and drought. The observed micromet data is passed through various scientific models to analyze the heritability of traits in pre-breeding and breeding. The resulting data is relevant to a number of clients, such as plant biologists, environmental researchers, plant breeders, farmers, and funding bodies. Client applications require different services from the sensor network and they query it to retrieve data. The sensor network services may vary configuration as the result of environmental changes or other interesting phenomena. We envision that a semantic-based publish/subscribe system, running in conjunction with the network and data stores, will match the right capabilities to client applications, allowing them to add and remove subscriptions dynamically as the service configurations change.



(a)



(b)

Fig. 2. Real time sensor observations: (a) Phenonet GUI; (b) sensed data.

Consider, for example, a standing query (a conjunctive query over terms and properties from the ontology) submitted by a client application that is checking for temperature threshold values. The query represents interest in the data underlying the queried concepts. In this case, the observed properties (Figure 2) are sensor locations, sensor accuracy and sensor manufacturer. If any change occurs in some part of the network or the ontology itself, on which the query is built, the query will be required to change as well. A client application would thus subscribe to concepts, properties and named individuals in the query. The system then monitors change to the ontology and service configuration and notifies subscribers of changes that logically or structurally affect them. The dependence on a change can be direct or indirect (implied), since the ontology is a logical object.

From the sensor network perspective, a fundamental example is when the network is configured and programmed automatically using an ontology [33], with the ontology treated as data loaded into a query interface. In this case, if terms in the ontology are moved or redefined, or if additions are made either to the ontology or the capabilities of the network, the query interface and its associated compilation procedure will be notified and the reconfiguration will be reflected in the client interface to take advantage of the changes.

An application service that allows the definition, and automated orchestration of new sensors and services in terms of the concepts (sensors and existing services) in the ontology [34] can work as part of the network configuration described above or by building compositions on top of the network's capabilities. If the composite sensor application service is defined as a concept in the ontology, the concepts and roles used in the definition and construction of a new sensor can subscribe to it. If change occurs in any part of the ontology or network, on which the new sensor depends, the composition service can be notified to reconstruct the composite sensor from the updated definitions and network capabilities.

5. Research Issues and Engineering Challenges

Several research and engineering challenges are associated with the lifecycle of a self-configuring a sensor network service. They lie in the observed measurements, in the construction of sensor service configurations, in the generation and selection of alternative configurations and action plans, and overall in the operational activities for ongoing adaptation during the lifetime of a sensor network. This lifecycle (Figure 3) begins with the design and implementation, proceeds to installation, configuration, monitoring, upgrading, and ceases in decommissioning of a sensor service.

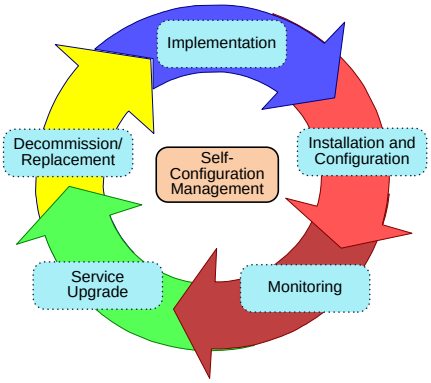


Fig. 3. Lifecycle of a self-configurable sensor network service.

Implementation: This phase involves service design, testing and verification. The runtime infrastructure should leverage distributed components to create an optimal configuration according to both application requirements and context. It should also allow uniform access to data by applications, irrespective of the data format, source or location. An ontology can be used to represent context and sensor capabilities.

Installing and configuring: This phase involves bringing up a service to an operational state with minimal user involvement. It requires explicitly stating a user subscription to correspond to a published configuration in a formal request language. To enable this, a sensor service will entail a bootstrapping process that begins with the configurations registering itself with the AE. The service may also interact with AE to discover other services and dependencies it needs to complete its initial configuration.

Monitoring: This phase is included in the self-configuration management, governed by AE. It involves monitoring configuration changes for sensor network services. This is an essential stage for enabling self-configuration ability for a service. When coupled with event correlation or decision theory, the monitored information is useful for problem identification and recovery during system faults.

Service upgrade: The self-configuring sensor network services may require upgrading them from time to time. They may subscribe/interact to an alert service that provides information on the availability of relevant upgrades and decide for themselves when to apply the upgrade, possibly with guidance from AE.

Decommission/replacement: Alternative to an upgrade process, sensor network services could be implemented afresh as part of a system upgrade, removing outdated services only after the new ones obtain a proper working status.

Lifecycle management: AE performs simultaneous activities to schedule and prioritize operations. A user interacts with a service via an appropriate interface provided by AE and retrieves the service description directly from the service. As per the node composition rule, the sensor service is configured and invoked according to application scenarios and context. During the lifetime of the system, services are dynamically reconfigured in response to sensed observations and system changes.

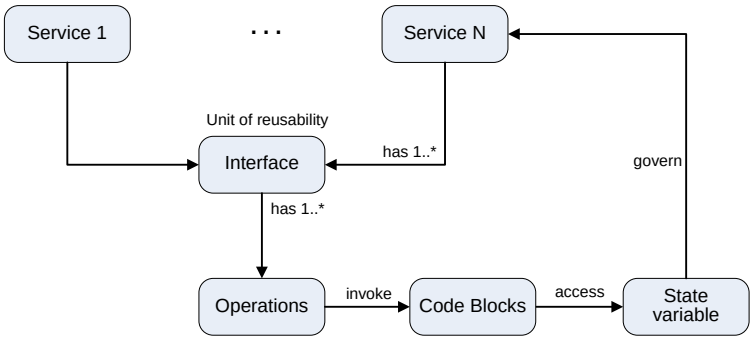


Fig. 4. Abstract model for a self-configurable sensor network service.

6. System Model

Figure 4 presents an abstract model for a self-configurable service. In our model, we formally capture the structure and run-time behavior of a sensor network service. The structure is defined by the service configurations and status in the system, whereas the run-time behavior is defined by the activities that execute service actions. We assume that the state of a sensor service is observable and that the results of actions influence the observables and state variables. Service types are specified in terms of one or more interfaces, whereby one interface may be attached to several configurations. An interface characterizes the behavior of a service and enables a common approach to represent the capabilities of a sensor service. We consider interfaces to be the unit of reusability. A self-configurable service is represented by a tuple **<*C, V, T*>**, where:

- ***C*** is the set of code blocks to perform service functionality $\Gamma \in \alpha \times \beta$, where α is the set of inputs and β is the set of outputs of a sensor network service and Γ defines a valid input-output set. A code block $c \in C$ makes use of state variables $v \in V$. Code blocks in the system are invoked through system operations, $t \in T$. One operation can invoke several code blocks.
- ***V*** is the set of state variables in the system, accessed through the code blocks. It can be expressed as $<\delta, \gamma>$, where δ is the set of sensors to provide services and γ is the set of constraints set that controls access to the sensor and its services.

Constraints are based on state and/or context, and can control who invokes the sensor service, when and how they are invoked and configured.

- ***T*** is the set of operations for self-configuration that are executed by sequentially invoking code blocks in the system, based on the system state changes by manipulating the system variables in *V*.

```
      begin
      ...
1:    initial system configuration
2:    for each service
3:     workload, usage pattern and system measurement
4:     if change identified = TRUE
5:       do construct and parameterize configurations
6:         if application context is met
7:           continue
8:         else if application context is not met
9:           explore alternative configuration
10:          evaluate and select best configuration
11:          put best configuration into effect
12:        else
13:          degrade service and report exception
14:        end if
15:     end if
16:   end for
      ...
      end
```

Fig. 5. Pseudo-code for the self-configuration procedure.

6.1. Self-Configuration Procedure

Figure 5 illustrates the procedure for self-configuration. Through online observations of state variables, e.g. sensor locations, accuracy, manufacturer, capability, service parameters are estimated. A change in the system condition or workload results in an alternation of parameters, after which the service status for those changed parameters are evaluated. It is determined whether a service under the changed condition meets the application context. If the current configuration is appropriate, nothing has to change (line 1-7). Otherwise, alternative configurations are explored and adaptive action plans are generated. Thereafter, the best alternative is selected and put into effect. In the face of failure, service is degraded with reporting exception (line 8-16).

7. Implementation Plan

To realize our architecture, we require methods to define service configurations, publish them for subscription and communicate with applications (Section 2.2). It calls for the use of the standards-based semantic technologies. Our approach involves expressing the sensor service overlay using a declarative language, such as Snlog, and executing the resulting specification to construct and maintain it. This approach is appealing, as it allows concise overlay specification, yet providing enough detail to be executed with performance and robustness acceptable to sensor applications [21].

As the cornerstone of the proposed architecture, we intend to build a semantic publish/subscribe component using an Enterprise Service Bus (ESB), such as WSO2. It provides fundamental services for complex architectures via an event-driven and standards-based messaging engine (the "bus"). The reasons for using an ESB are increased flexibility, decentralization, faster integration to existing systems, and distributed deployment. WSO2 ESB provides a built-in repository to store service configurations and metadata. A significant feature of the semantic publish/subscribe component will be its ability to support flexible subscriptions.

A part of the central registry module (Figure 1) will represent the complete sensor network configuration modeled using the W3C SSN-XG sensor ontology, expressed in the Web Ontology Language (OWL). By computing the ontology subsumption hierarchy with a reasoner, the registry, under governance of AE, will generate a topic hierarchy of classes and properties that is made available for subscription.

Client applications are generally interested, and dependent on, only a portion of the complete sensor network ontology. For initialization, a client application may submit a number of conjunctive queries to AE to obtain sensor configuration data from the registry. Alternatively, the client may describe its interest as a reference to a module of the ontology. From these requests, the interest of the client can be summarized as the classes and properties mentioned in the query, or the highest classes and properties in the module, respectively. This interest is reflected as a subscription to the named topics managed in the WSO2 ESB topic hierarchy—future configuration changes to the network will be published on this topic channel but pre-existing messages should be discarded by the client.

When the central registry is updated to reflect a change in the underlying sensor network configuration, client applications will be informed of changes of interest to them through messages generated and labeled according to the topic hierarchy. In this case, differences in the updated ontology have been computed by AE and the corresponding set of affected topics is extracted as classes and properties mentioned in the difference list. It is not straightforward to compute the difference between ontology versions in an expressive ontology such as those in OWL. In some cases, database-style tracking of updates over ABox instances is sufficient. More generally, we intend to use methods for discovering the logical succinct difference between two ontology versions [15-16], Although not yet solved for all description logics (and possibly not decidable for some), it is a promising new method to automatically compute the difference in the implications between two DL ontologies.

In our context, for TBox changes, we will cautiously overestimate the impact of changes. Specifically, we intend to compute exact or overestimated changes, thus also notify additional subscribers who need not have been notified. As a result, we prevent

leaving subscribers with an inconsistent view of the system and ontology state, due to the errors in change estimations. A client application only receives notification from AE when the updated configuration corresponds to a topic to which it subscribes. A subscriber then responds by resubmitting its original request for configuration data from the registry, and recompile, reconfigure or reload as appropriate for its context.

While we envisage that the use of semantics will greatly improve the power and flexibility of the matching process, one possible issue could be the overhead caused from the ontology update, thus leading to the computation of logical difference and new subsumption hierarchy creation. However, given the fact that the topic hierarchy changes infrequently, the overhead caused from ontology update is not a major concern. On the other hand, ontology changes confined to the ABox may occur more frequently in practice but are relatively easily computed.

8. Conclusion and Future Work

We have presented an architecture to enable self-configuration of sensor network services. A semantics-based publish/subscribe approach, based on a plug-and-play concept, has been employed to assist the self-configuration process. The architectural approach will offer the required level of abstraction and generality to integrate to an existing system. A list of associated research challenges has also been identified to proceed with the practical implementation. We intend to implement the approach presented in this paper within a real-world sensor network deployment, such as the Phenonet project. Our work relates to the intelligence of a sensor network system to operate autonomously with minimal user intervention in different application scenarios and context. We envisage that this semantics-based approach will not only provide a conceptual foundation for self-awareness in sensor networks, but also will allow a sensor service to describe, use and adapt its behavior to its current context with the use of semantic technologies. Our future work will proceed in this direction.

Acknowledgement

We thank the members of the Phenonet project, in particular, Doug Palmer, Peter Lamb, Robert Furbank, Xavier Sirault, Leakha Henry, and Christopher Swain. This work is funded in part by National Collaborative Research Infrastructure Strategy (NCRIS) and the High Resolution Plant Phenomics Centre (HRPPC), Australia.

References

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE communications magazine,* vol. 40, no. 8, pp. 102-114, 2002.

[2] R. Balani, C. C. Han, R. K. Rengaswamy, I. Tsigkogiannis, and M. Srivastava, "Multi-level software reconfiguration for sensor networks," *Proc. International Conference on Embedded Software (EMSOFT'06),* pp. 121-130, ACM Press, NY, USA, 2006.

[3] M. Botts, G. Percivall, C. Reed, and J. Davidson, "OGC® sensor web enablement: Overview and high level architecture," *GeoSensor Networks,* pp. 175-190, 2008.
[4] A. Bröring, K. Janowicz, C. Stasch, and W. Kuhn, "Semantic challenges for sensor plug and play," *Web and Wireless Geographical Information Systems, LNCS,* vol. 5886, pp. 72-86, 2009.
[5] N. Bulusu, J. Heidemann, D. Estrin, and T. Tran, "Self-configuring localization systems: Design and experimental evaluation," *ACM Transactions on Embedded Computing Systems (TECS),* vol. 3, no. 1, pp. 24-60, 2004.
[6] D. Chu, L. Popa, A. Tavakoli, J. M. Hellerstein, P. Levis, S. Shenker, and I. Stoica, "The design and implementation of a declarative sensor network system," *Proc. 5th International Conference on Embeded Networked Sensor Systems (SenSys'07),* p. 188, ACM Press, NY, USA, 2007.
[7] L. Clare, G. Pottie, and J. Agre, "Self-organizing distributed sensor networks," *The Proceedings of SPIE,* vol. 3713, no. 229, pp. 229–237, 1999.
[8] M. Compton, C. Henson, H. Neuhaus, L. Lefort, and A. Sheth, "A survey of the semantic specification of sensors," *Proc. 2nd International Workshop on Semantic Sensor Networks (SSN'09),* 2009.
[9] F. Facca, S. Komazec, and M. Zaremba, "Towards a semantic enabled middleware for publish/subscribe applications," *Proc. IEEE International Conference on Semantic Computing (ICSC'08),* pp. 498-503, IEEE CS Press, Los Alamitos, CA, USA, 2008.
[10] B. Furbank, X. Sirault, and D. Deery, "Phenonet: A distributed sensor network for field crop phenotyping," *Proc. 1st International Plant Phenomics Symposium : from Gene to Form and Function,* 2009.
[11] P. Hitzler, M. Krötzsch, B. Parsia, P. Patel-Schneider, and S. Rudolph, "OWL 2 Web ontology language primer," *W3C Recommendation,* http://www.w3.org/TR/2009/REC-owl2-primer-20091027, 2009.
[12] IBM, "An architectural blueprint for autonomic computing," *White Paper,* June 2005.
[13] J. Jeong and D. Culler, "Incremental network programming for wireless sensors," *Proc. 1st Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON'04),* pp. 25-33, 2004.
[14] M. Klopfer and I. Simonis, *SANY-an open service architecture for sensor networks*: SANY Consortium, 2009.
[15] B. Konev, D. Walther, and F. Wolter, "The logical difference problem for description logic terminologies," *Automated Reasoning, Lecture Notes in Computer Science,* vol. 5195, pp. 259-274, 2008.
[16] R. Kontchakov, F. Wolter, and M. Zakharyaschev, "Can you tell the difference between DL-Lite ontologies," *Proc. 11th International Conference on Principles of Knowledge Representation and Reasoning,* pp. 285–295, 2008.
[17] J. Kramer and J. Magee, "Self-managed systems: An architectural challenge," *Proc. International Conference on Future of Software Engineering (FOSE'07),* pp. 259-268, 2007.
[18] L. Li and K. Taylor, "A framework for semantic sensor network services," *Proc. International Conference on Service Oriented Computing (ICSOC'08),* pp. 347-361, 2008.
[19] L. Li and K. Taylor, "Generating an efficient sensor network program by partial deduction," *Proc. 11th Pacific Rim International Confernece on Artificial Intelligence (PRICAI'10),* 2010.
[20] A. Lim, "Distributed services for information dissemination in self-organizing sensor networks," *Journal of the Franklin Institute,* vol. 338, no. 6, pp. 707-727, 2001.
[21] B. T. Loo, T. Condie, J. M. Hellerstein, P. Maniatis, T. Roscoe, and I. Stoica, "Implementing declarative overlays," *ACM SIGOPS Operating Systems Review,* vol. 39, no. 5, p. 90, 2005.

[22] D. Marsh, R. Tynan, D. O'Kane, and G. P O'Hare, "Autonomic wireless sensor networks," *Engineering Applications of Artificial Intelligence,* vol. 17, no. 7, pp. 741-748, 2004.
[23] L. Mottola and G. P. Picco, "Programming wireless sensor networks: Fundamental concepts and state of the art," *ACM Computing Surveys,* 2010. To Appear.
[24] M. Petrovic, I. Burcea, and H. Jacobsen, "S-ToPSS: Semantic Toronto publish/subscribe system," *Proc. 29th VLDB Conference,* pp. 1104-1107, 2003.
[25] M. Presser, P. Daras, N. Baker, S. Karnouskos, A. Gluhak, S. Krco, C. Diaz, I. Verbauwhede, S. Naqvi, F. Alvarez, and A. Fernandez-Cuesta, "Real World Internet," *Position Paper,* Future Internet Assembly, 2010.
[26] J. Qian, J. Yin, D. Shi, and J. Dong, "Exploring a semantic publish/subscribe middleware for event-based SOA," *Proc. IEEE Asia-Pacific Services Computing Conference,* pp. 1269-1275, IEEE CS Press, Los Alamitos, CA, USA, 2008.
[27] D. Roggen, K. Förster, A. Calatroni, T. Holleczek, Y. Fang, G. Tröster, P. Lukowicz, G. Pirkl, D. Bannach, and K. Kunze, "OPPORTUNITY: Towards opportunistic activity and context recognition systems," *Proc. Third IEEE WoWMoM Workshop on Autonomic and Opportunistic Communications,* 2009.
[28] A. Sheth, C. Henson, and S. Sahoo, "Semantic sensor web," *IEEE Internet Computing,* vol. 12, no. 4, pp. 78-83, 2008.
[29] R. Spalazzese, P. Inverardi, and V. Issarny, "A formalization of mediating connectors: Towards on the fly interoperability," *Technical Report,* TRCS 004/2009, University of L'Aquila, 2009.
[30] T. Stathopoulos, J. Heidemann, and D. Estrin, "A remote code update mechanism for wireless sensor networks," *Technical Report,* CENS-TR-30, University of California, 2003.
[31] M. Stollberg, "SHAPE: Service and Software Architectures, Infrastructures and Engineering," *White Paper,* 2010.
[32] R. Sugihara and R. K. Gupta, "Programming models for sensor networks: A survey," *ACM Transactions on Sensor Networks (TOSN),* vol. 4, no. 2, pp. 1-29, 2008.
[33] K. Taylor and P. Penkala, "Using Explicit Semantic Representations for User Programming of Sensor Devices," *Proc. Australasian Ontology Workshop,* 2009.
[34] K.-N. D. Tran, "Semantic sensor composition," *BSc Thesis,* Australian National University, http://cs.anu.edu.au/people/Nguyen.Tran/SemanticSensorComposition/Thesis/thesis_honours.pdf, 2009.
[35] M. Uschold, P. Clark, F. Dickey, C. Fung, S. Smith, S. Uczekaj, M. Wilke, S. Bechhofer, and I. Horrocks, "A semantic infosphere," *Proc. International Semantic Web Conference (ISWC'03),* pp. 882-896, 2003.
[36] T. Usländer, "Reference Model for the ORCHESTRA Architecture (RM-OA)," *Open Geospatial Consortium https://portal. opengeospatial. org/files,* 2005.
[37] J. Wang, B. Jin, and J. Li, "An ontology-based publish/subscribe system," *Proc. Middleware'04,* pp. 232-253, 2004.
[38] B. Williams and P. Nayak, "A model-based approach to reactive self-configuring systems," *Proc. AAAI'96,* pp. 971-978, 1996.
[39] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer networks,* vol. 52, no. 12, pp. 2292-2330, 2008.
[40] L. Zeng and H. Lei, "A semantic publish/subscribe system," *Proc. IEEE International Conference on E-Commerce Technology for Dynamic E-Business (CEC-East'04),* pp. 32-39, IEEE CS Press, Los Alamitos, CA, USA, 2004.
[41] W. Zhang and K. Hansen, "Semantic web based self-management for a pervasive service middleware," *Proc. Second IEEE International Conference on Self-Adaptive and Self-Organizing Systems,* pp. 245-254, IEEE CS Press, Los Alamitos, CA, USA, 2008.