Using Dynamic Software Variability to Manage Wireless Sensor and Actuator Networks

Mary Luz Mouronte
Departamento de Ingeniería y Arquitecturas Telemáticas
Universidad Politécnica de Madrid, Spain
mouronte.lopez@upm.es

Óscar Ortiz
Departamento de Ingeniería y Arquitecturas Telemáticas
Universidad Politécnica de Madrid, Spain
oscar.ortiz@upm.es

Ana Belén García
Departamento de Ingeniería y Arquitecturas Telemáticas
Universidad Politécnica de Madrid, Spain
anabelen.garcia@upm.es

Rafael Capilla
Departamento de Ciencias de la Computación
Universidad Rey Juan Carlos
Madrid, Spain
rafael.capilla@urjc.es

***Abstract*— This paper explores the use of dynamic software variability techniques to manage Wireless Sensor and Actuator Networks (WSANs), and we describe both an architecture and a dynamic software variability mechanism that can be integrated in WSAN scenarios. Our main goal is to apply runtime variability to describe the context-aware properties of a WSAN family and manage this more efficiently by supporting dynamic changes in the structural variability that can be reflected in the configuration of the WSAN at runtime. Our approach contributes to increase self-management of WSANs based on dynamic configurable system options.**

Keywords—Wireless Sensor and Actuator Network management; Dynamic software product lines; Runtime variability; Context-aware networks.

I. INTRODUCTION

Nowadays, the ever increasing complexity and the incessant demand of context-aware systems that need to react to different context conditions dynamically and on demand, drive the necessity to handle the management of runtime changes. Hence, context information constitutes an important source of data for those systems that have to react dynamically to changes in the environment or to new context conditions. Wireless Sensor and Actuator Networks (WSANs) constitute a clear example of this type of systems.

WSANs, broadly speaking, can be found in different environments, from mobile systems and wearable computing to WSANs composed of fixed nodes deployed either in a deterministic or random manner, to name a few. All of them have the necessity of considering a diversity of runtime scenarios and optimizing their operation in consistence with the contextual information. Furthermore, this dynamic handling of the context has to be done, for the most part, autonomously (self-* capabilities) and robustly.

Challenges of this approach are manifold. To give some examples, WSANs may exhibit energy scarcity and thus having complex software running in the nodes or extensively using the wireless interfaces (e.g. to upload new software to the nodes) are things to be avoided. Also, most of the functions offered by a WSAN (taken as a product) are implemented in a distributed manner, which greatly affects the actual management procedures and technologies to be used. In summary, all these circumstances are adequate to find an effective and formalized way for bridging the gap between the software-engineering-related variability models and the network-related WSAN management. Any step towards this unified management would greatly ease the manageability of context-aware systems that need to evolve dynamically. The main contribution of this position paper is a high-level architecture that exploits runtime software variability techniques to manage dynamically the context-aware information of a WSAN.

The remainder of this paper is as follows. Section II describes the related work. In section III we describe our approach, which continues and builds on a previous work, to design a WSAN system family that uses runtime variability to manage dynamic changes more efficiently. Section IV outlines a motivating example of how our proposed architecture is applied. Finally, in section V we draw the conclusions and outline the future work.

II. RELATED WORK

A. WSAN systems

There are several works about how to get systems and services to manage themselves (autonomic management), by introducing autonomic computing mechanisms into network and services management [1]. More specifically related to WSAN management, [2] describes the design of an open source management system as an extension of OpenRSM to support TinyOS facilities. Regarding the reconfiguration of WSANs, recent works suggest a component-based distributed framework that allows to make changes only in specific components, thus saving energy ([3], [4]). Similarly, the authors in [5] propose a mobile multi-agent architecture where individual agents may be reconfigured or uploaded to nodes. These works provide a good basis for building a part of our model, although they lack the description of a systematic and formalized relationship between the functionalities perceived by the user of the WSAN as a product and the specific reconfiguration changes to be triggered either to or from the network.

B. Runtime Variability

Software variability is a technique used in the Software Product Line [6] area and successfully applied in industry for more than 20 years, and used to describe the variations of a set of related systems of behalf of a set of variants [7] or system features that are organized around a Feature Model (FM). The variants in the system are organized in variation points that describe the logical relationships between variants and they can bind to concrete values at different binding times to realize such system options. At present, there are few proposals in the literature suggesting the use of software variability to describe WSAN properties. Some of these are focused on specific application domains such as autonomic smart homes [8], while others focus on tools aimed to support the feature model of a WSAN [9]. In addition, the authors in [10] extend a feature model with contextual parameters to trigger reconfiguration.

However, none of the existing proposals suggests paths to modify the properties of a WSAN dynamically using a variability model. In this context, recent research in the product line area suggest the notion of Dynamic Software Product Lines (DSPLs) [11] able to manage the dynamic changes of context-aware systems, and exploit runtime variability mechanism [12] [13] to support the activation and deactivation of system options at runtime and handle those system features specific to context properties.

III. Dynamic Management of Wireless Sensor and Actuator Networks: Architectural Proposal

A. Background

This research work builds on a recent work [14] that describes how wireless sensor and actuator networks can be seen as a product line, as we modeled the variability of a family of WSAN using feature models. The novelty of this work relies on the dual software architecture shown in Figure 1 that manages the properties of a WSAN dynamically and reflect these changes at runtime to the software designer, who can also apply changes in the structure of the WSAN using a runtime variability model. Facilities for self-management are implemented using a runtime variability mechanism able to support the changes in the WSAN context properties and devices as well, where new WSAN properties and devices can be added, changed, or removed dynamically.

B. Overview of the Proposed Architecture

The proposed architecture of Figure 1 is naturally distributed respecting the WSAN. In fact, a distributed component software or middleware will be generally deployed in the network nodes, and the distributed nature of many of the features implementation is considered. What we propose is to use a "centralized subsystem" that contains both reasoning engines and databases in order to manage the variable system options of the WSANs, i.e., to introduce the concepts and mechanisms of DSPL disciplines into the WSAN arena. The architecture composed by two main parts:

(i) the left part of the figure manages the runtime variability model and the Configuration Database (CfDB) for supporting the variable options associate to context properties of the WSAN products. Any change in the FM (e.g. activating/deactivating or adding/removing system features) will be consistently updated in the CfDB. The FM also includes constraint rules that are checked to avoid incompatible product configurations, thus any change in the WSAN feature model has to be validated by a Constraints Validation Engine (CrVE) before committing it to the CfDB. Reciprocally, any change made to the CfDB that originates in the WSAN scope (see below) will be reflected in the feature model.

(ii) the right part of the figure manages the configuration of the WSAN properties described in the CfDB database, and handles the activation and deactivation of WSAN context properties. The WSAN network is composed by ubiquitous devices (static, mobile, or even wearable) able to perform the following tasks:

- Sensors (e.g. temperature, humidity, or heart-rate sensors) measure environmental parameters and send them to the WSAN Manager – this may involve multi-hop communication.
- Actors (e.g. alarms, HVAC systems) carry out actions upon the environment by means of actuators.

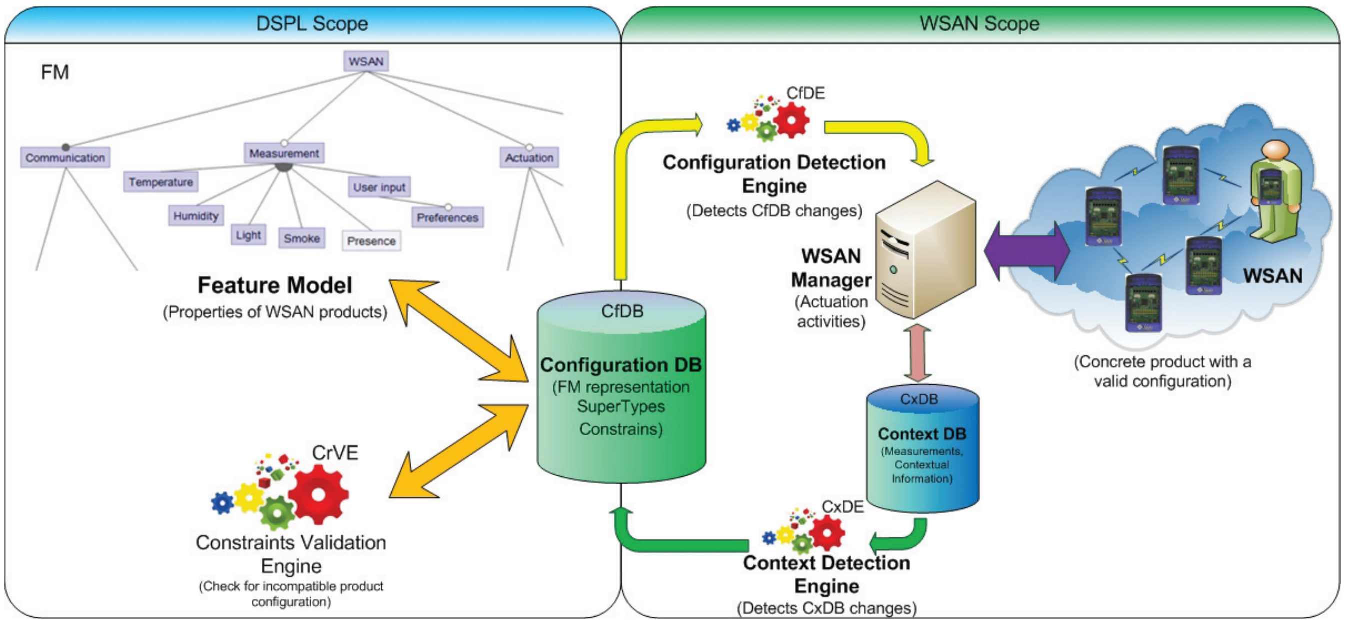

Fig. 1. Architecture of the proposed WSAN dynamic management system.

In Figure 1, the WSAN Manager is responsible of gathering the measurements and storing them inside the Context Database (CxDB). It also may trigger configuration changes to the WSAN nodes, and command them to perform actuation activities.

In addition, a Context Detection Engine (CxDE) uses the information stored in the Context Database (CxDB) and performs the necessary changes to the Configuration Database (CfDB). This engine consists of a process, e.g. a daemon, which continuously monitors the CxDB for the appearance of something that would trigger the activation of the corresponding feature, e.g. the appearance of measurements of a new physical magnitude. Likewise, the Configuration Detection Engine (CfDE) will detect any change in the CfDB and instructs the WSAN Manager to send the corresponding commands to the nodes for the new required configuration.

In addition, when a new WSAN configuration is required (e.g. measure a new parameter), the corresponding variant in the feature model must be activated or added dynamically, and such change must be validated and reflected in the CfDB database. Afterwards, the CfDE will detect such update and the WSAN Manager will enact the appropriate tasks to update the nodes of the network. Reciprocally, any contextual information generated in the network and stored in the CxDB, which, according to the CxDE detection rules, demands a configuration change, will be updated in the CfDB. This will be done, as always, if the CrVE validates this configuration change.

C. Representation of the Variability: Feature Model

In order to manage the WSAN features that will be activated or modified dynamically, we used software variability techniques and in particular a feature model to outline the visible characteristics of our WSAN products. In our approach, we highlight those features that handle contextual information as we believe are those that change more often at runtime.

The feature model, shown partially in Fig 1, is based on our previous work [13] [14], where we identified four major feature groups of a WSAN product family, related to: *Communication, Measurement, Actuation,* and *Localization* properties. For instance, the *Measurement* feature group encompasses a set of configurable and context-aware variants able to handle *temperature, humidity, smoke*, and other context features of the WSAN product. Some of these features are optional (e.g.: user generated alarm) while others are mandatory for all WSAN products. Our runtime variability model uses a taxonomy to classify system properties in what we call Super-Types (STs). Therefore, each super-type encompasses a common functionality for supporting the WSAN properties and configurable options (e.g.: the measuring system options), and provides a way to add, change, or remove properties in the variability model using the super-types.

D. Configuration and Context Databases

The Configuration Database (CfDB) contains all the information related to the current and potential functionality of the product. That is, it contains a representation of the FM. The CfDB also stores the set of constraints that have to be met for any product.

The Context Database (CxDB) can be seen as the "output" of the actual product that is currently configured. Strictly speaking, it is related to one part of this output: the contextual information that is being sensed by the WSAN. The other part, which contains the possible actuation activities of the WSAN, is instrumented by the WSAN Manager, either after a change in the configuration detected by the CfDE or by some kind of internal logic related to the application (e.g. "if average temperature drops below a threshold, then turn the HVAC system on").

Thus, CfDB and CxDB store information that is different both in nature and dynamicity, although the Detection Engines make them related to each other. We are using an off-the-shelf RDBMS (concretely, MySQL) to design and implement both databases. It is important to keep in mind that it may be necessary to prune or summarize older contextual information stored in CxDB, especially in production stages, in order to enhance scalability, due to the highly dynamic and ever-growing nature of the information stored in this database.

E. Detection and Validation Engines

There are two Detection Engines (Configuration DE and Context DE) that together implement the correspondence between the two databases and support the connection between the variability model and the contextual data gathered by the WSAN.

F. Management of the Context-aware Network

As mentioned before, in a WSAN network, sensors collect data about the physical world, while actors decide and carry out suitable actions upon the environment. An actor interacts with the environment by means of actuators, which are devices that transform an electrical control signal into a physical task, i.e. actuators establish a mechanism by which an actor acts upon the physical environment. Hence, a **N**etwork **M**anagement **S**ystem (NMS) is needed to support the variability related to the physical environment, which supervises data from the environment and controls actors remotely.

In our approach we suggest the NAGIOS open source platform (http://www.nagios.org) for building the WSAN management mechanism, and monitoring the WSAN. Hence, any problem detected in the network will be notified to NAGIOS, which can also extract and analyze numerical data gathered by the sensors. The results received by NAGIOS are collected in the CxDB for triggering the updates to the CfDB. Finally, those changes committed to the CfDB provide, via the CfDE, will be managed by NAGIOS in the WSAN.

IV. MOTIVATING EXAMPLE

As a motivating example, let's assume that a new variant called "Smoke" is added to the CfDB. This variant belongs to the "Ambient" and "Safety" Super-Types, and contains two properties: a period with which the nodes have to send the measurements (related to "Ambient"), and a threshold above which the detecting node has to immediately notify the manager (related to "Safety"). The CfDE will detect this and notify the WSAN Manager, which will instruct all nodes with smoke-detection capabilities to activate the periodical measurements and notify whenever the threshold is trespassed. Consequently, the CxDB will start to be populated with smoke measurements and possibly smoke alarms.

In addition, in case the WSAN network incorporates a new feature causing changes in the information stored in the CxDB (e.g. a new type of periodic measurement), the CxDE must decide if such new feature will be added and/or activated according to its Super-Type in the CfDB and the feature model. Example of similar scenarios can be found in our preliminary work [14].

Moreover, the Constraints Validation Engine (CrVE) intercepts any request to change the CfDB in order to check whether all constraints would be still met when applied to the modified features. For instance, the activation of a feature pertaining to the "Safety" ST requires that the feature "Event-based" (of the "Comm type" variation point) is also activated, since alarms of any type are basically events triggered by nodes. Ideally, a roll-back mechanism could be implemented to revert to a previous state of the CfDB database. To begin with, in early prototypes a simpler notification of the existence of an inconsistence will suffice.

V. Conclusions and Future Work

In this research we have described an architecture that applies the dynamic variability to manage WSANs and enables reconfiguration activities at runtime. Our approach automates the configuration and further changes to WSAN nodes and reflects the changes that happen in the WSAN to the properties described using feature models, which helps WSAN software engineers to understand what is happening in the network. Currently, we are aware of the lack of a proof of concept to test the proposed architecture but in order to support different WSAN scenarios, we have started the development of two prototype tools that will share a common database. The first prototype will support the runtime changes in the variability model (i.e.: we plan to use the FAMILIAR tool to support the variability model - https://nyx.unice.fr/projects/familiar/) that will be propagated in the WSAN. The second prototype will be built based on NAGIOS (URL), and will be in charge of managing the WSAN properties and storing the information received from the network in order for it to be reflected in the variability model.

Acknowledgments

This work has been partially funded by the Spanish Ministry of Economy and Competitiveness (Ministerio de Economía y Competitividad), in the framework of the project "Accessible wearable device platform for smart environments" (Ref. TEC2011-28397). Óscar Ortiz and Ana Belén García are members of the R&D centre "Centro de Investigación en Tecnologías Software y Sistemas Multimedia para la Sostenibilidad (CITSEM)" and of the Department of Telematic Engineering and Architectures (Departamento de Ingeniería y Arquitecturas Telemáticas – DIATEL, UPM).

References

[1] J. Serrat, A. Astorga, and J. Rubio, "Autonomic systems in network and service management", Upgrade, vol 9, No 6, 2008, pp.41-48.

[2] M. Kalochristianakis, V. Gkamas, G. Mylonas, S. Nikoletseas, E. Varvarigos, and J. Rolim, "An open and integrated management platform for wireless sensor networks", International Symposium on Autonomous Decentralized Systems, 2009, pp.1-6.

[3] D. Hughes, et al., "LooCI: A Loosely-Coupled Component Infrastructure for Networked Embedded Systems", 11th IEEE International Symposium on Network Computing and Applications (NCA), 2012, pp.236-243.

[4] A. Taherkordi, F. Loiret, R. Rouvoy, and F. Eliassen, "Optimizing Sensor Network Reprogramming Via in-Situ Reconfigurable Components", ACM Transactions on Sensor Networks, vol. 9, No.2, 2013, pp 1-37. Available online: http://hal.inria.fr/docs/00/65/87/48/PDF/RemoWare_TOSN.pdf, [accessed Jan. 2013]

[5] A. Santos, G. B. Nunes, P. Gil, and A. Cardoso, "Multi-Agent Platform in WSAN Applications: A Time Synchronization Perspective", 5th International Conference on Management and Control of Production and Logistics, 2010, pp. 367-374.

[6] K. Pohl, G. Böckle, F. van der Linden, "Software Product Line Engineering Foundations, Principles, and Techniques". Springer Verlag, 2005.

[7] K. Kang, S. Cohen, J. Hess, W. Novak, and A. Peterson, "Feature-Oriented Domain Analysis (FODA) Feasibility Study," Software Engineering Institute, Carnegie Mellon University, Tech. Rep. CMU/SEI-90-TR-21, 1990.

[8] C. Cetina, Pau Giner, J. Fons, and V. Pelechano, "Autonomic Computing Through Reuse of Variability Models at Runtime: The Case of Smart Homes", IEEE Computer vol. 42, No. 10, 2009, pp. 37-43.

[9] P. Boonma and J. Suzuki, "Model-driven performance engineering for wireless sensor networks with feature modeling and event calculus", In Proceedings of the 3rd workshop on Biologically inspired algorithms for distributed systems (BADS '11), ACM, New York, NY, USA, 2011, pp. 17-24.

[10] N. Gamez, L. Fuentes, and M. Aragüez, "Autonomic Computing Driven by Feature Models and Architecture in FamiWare", Lecture Notes in Computer Science, vol. 6903, 2011.

[11] S. Hallsteinsen, M. Hinchey, S. Park and K. Schmid, "Dynamic Software Product Lines", IEEE Computer, vol. 41, No. 4, 2008, pp. 93-95.

[12] R. Capilla, J. Bosch. "The Promise and Challenge of Runtime Variability". IEEE Computer vol., 44, No. 12, 2011, pp. 93-95.

[13] J. Bosch, R. Capilla. "Dynamic Variability in Software-intensive Embedded System Families". IEEE Computer, vol. 45, No. 10, 2012, pp. 28-35.

[14] O. Ortiz, A.B. García, R. Capilla, J. Bosch, M. Hinchey. "Runtime Variability for Dynamic Reconfiguration in Wireless Sensor Networks Product Lines", 6th DSPL Workshop (SPLC), 2012, pp. 143-150.