Invited Poster:
makeSense: Easy Programming of Integrated Wireless Sensor Networks

Fabio Casati⋆, Florian Daniel⋆, Adam Dunkels†, Stamatis Karnouskos+, Patricio Moreno Montero∗,
Luca Mottola†, Felix Jonathan Oppermann‡, Gian Pietro Picco⋆, Kay Römer‡, Patrik Spieß+,
Stefano Tranquillini⋆, Paolo Valleri⋆, Thiemo Voigt†
†Swedish Insitute of Computer Science, ‡University of Lübeck (Germany),
∗ACCIONA Infraestructuras (Spain), +SAP Research (Germany), ⋆University of Trento (Italy)
Contact e-mail: `thiemo@sics.se`

***Abstract*— WSNs are expected to play a critical role in the next computing revolution, as depicted in the visions of Cooperating Objects and the Internet of Things. However, designing and developing WSN software is currently very difficult. This may prevent WSNs from reaching large-scale adoption, especially in industry. The makeSense project aims at enabling an easier integration of WSNs in business processes, by allowing business process experts and WSN developers to express the high-level functionality required, while leaving low-level details to the compiler and run-time system. We envision the results of makeSense to be not only a landmark for WSN software development, but also a new way to look at WSN programming that increases productivity and business value, enabling a far-reaching adoption in key industrial domains.**

I. Introduction

Wireless Sensor Networks (WSNs) are a key component towards the integration of the physical and virtual worlds, as depicted in the visions of Cooperating Objects and Internet of Things. However, their widespread adoption in industry is still limited. Factors that hinder a more widespread adoption are the difficulty of WSN programming, as acknowledged in the CONET research roadmap [2], and the limited support for integration with existing IT infrastructures. In particular, although several programming abstractions are available in the literature [3], almost none of them explicitly supports the integration of WSNs with business processes.

The EU-funded make*Sense* project enables such integration by devising programming abstractions to express the high-level WSN functionality within existing business process modeling concepts. This allows for seamless specification of the behavior of the WSN and the surrounding business process. Low-level details are then left to a dedicated compiler and run-time system. The name, make*Sense*, reflects both purpose and ambitions of the project. The first part of the name, `make`, refers to the `make` tool, the software development utility that relieves developers of software development details.

Section II of this paper illustrates the make*Sense* approach and the overall architecture. Section III elaborates on the expected results of the project and concludes.

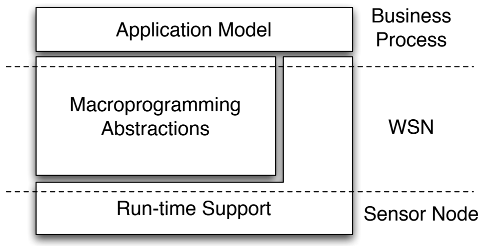

Fig. 1: make*Sense* architecture.

II. Approach

Consider an automatic building ventilation system that integrates with an on-line meeting room reservation application. The current approach is to manually ventilate the room either at fixed intervals irrespective of any meetings, or to trigger the ventilation manually, e.g., by the meeting participants. Smart control strategies for ventilation systems may allow to adjust the ventilation levels to the actual demand, based on a room's scheduled and monitored occupancy, while assuring adequate environmental quality. This can save up to 30% of the energy used for air conditioning in a building. The latter account for more than 40% of the energy consumption in Europe.

The smart control system uses WSNs to check the presence of people and the CO_2 levels in rooms. The CO_2 monitoring starts 15 minutes prior to a meeting. If CO_2 is above a specific threshold, the system automatically triggers the ventilation. This check continues periodically. Additionally, 15 minutes after the scheduled start of the meeting, the system starts monitoring the presence of persons. If presence is detected, the system updates the status of the room in the reservation application to "occupied"; otherwise, the room status is set to "available" and the periodic monitoring of CO_2 stops.

To ease the design and implementation of applications such as the one described above, make*Sense* follows an approach consisting of three layers, as depicted in Figure 1:

- The *application model* layer integrates sensor networks with business application systems by allowing WSN behavior to be expressed within a business process model.
- The *macro-programming* layer provides a network-centric

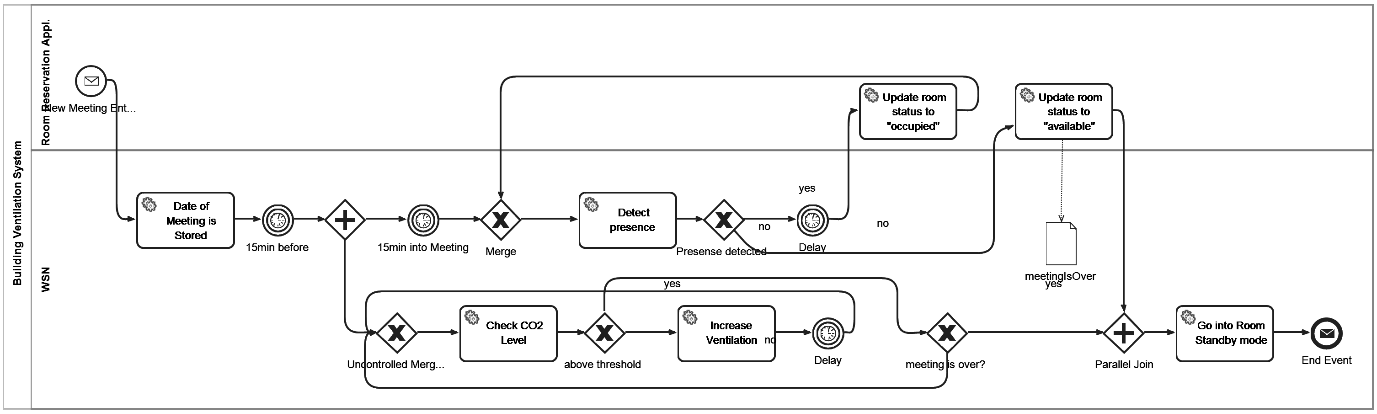

Fig. 2: Meeting room ventilation system: overall business process.

- programming abstraction that relieves programmers from the low-level details, also allowing existing and future abstractions to blend smoothly.
- The *self-optimizing run-time system* layer adapts to the specific conditions in the deployed sensor network by optimizing communication and resource consumption based on inputs from the higher layers.

In make*Sense*, we use BPMN (Business Process Modelling Notation) at the application model layer, as this is the de-facto way in industry for specifying business processes. Moreover, BPMN is today integrated in the majority of tools used by business process developers, and compliance with such tools may foster a quicker adoption of make*Sense*.

Figure 2 depicts the BPMN specification of the meeting room ventilation system. Through a refinement step, developers specify the WSN processing to carry out the activity “Detect presence” and “Check CO_2 Level”. These WSN-specific functionality are expressed using custom BPMN constructs that may include constraints on sensed data, e.g., the conditions to detect the presence of persons.

The BPMN specification is used as input by the make*Sense* model compiler, along with the network model and high-level performance objectives. The former includes information on the application-level characteristics of the deployed nodes, e.g., their capabilities and logical location (e.g., “room ABC”). The latter express preferences on possibly conflicting performance goals; for example, to optimize energy consumption at the cost of increased latency.

Based on this information, the compiler outputs a macro-program that describes the WSN processing using high-level, network-wide programming constructs. For instance, the macro-code required for CO_2 monitoring looks like:

```
co2sensors ::= Type = ``sensor'' AND Function = ``CO2''
controller ::= {count (room) { role == controller } == 0}

when [co2sensors] report [suddenIncrease]
tell [controller] to [increaseVentilation]
```

The fragment of code above determines a subset of nodes responsible for sensing the CO_2 levels, and elects one controller per room in charge of triggering the actuation when necessary. The language leverages existing WSN abstractions, Logical Neighborhoods [3] and Generic Role Assignment [3] in this case, by integrating them seamlessly in the same programming framework. The macro-program is then input to a macro-compiler, along with the network model. The macro-compiler translates the network-wide program in executable node-level code, depending on the nodes capabilities and role.

The executable code runs atop a dedicated run-time layer, based on the Contiki [1] operating system. In addition to enabling dynamic reconfiguration of the deployed functionality, the run-time layer continuously runs a monitoring and self-optimization process. This provides feedback to the application model layer on the current system performance, and adapts the system operation based on the user-defined performance objectives and the current network conditions.

III. Expected Results and Conclusion

The core contribution of make*Sense* is a comprehensive programming system that enables the integration of WSNs and business processes. The programming platform is supported by a complete tool-chain starting from business processes down to the code running on the individual nodes, including tools to assist the developer in the programming activity and the compiler technology required to bridge the gap between the (business) application level and the WSN hardware.

We plan to evaluate the results in a real-world scenario. One possibility is precisely to deploy a prototype system developed with make*Sense* in the context of a building ventilation system. To this end, we will introduce wireless sensors performing real-time metering of relevant environmental parameters and the current energy consumption of the building. We will then qualitatively and quantitatively assess the developers’ productivity as well as individual component and overall system performance.

References

[1] A. Dunkels, B. Grönvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Proc. of the Workshop on Embedded Networked Sensor Systems (Emnets)*, 2004.

[2] P. J. Marrón, S. Karnouskos, D. Minder, and A. Ollero, editors. *The emerging domain of Cooperating Objects*. Springer, 2011.

[3] L. Mottola and G. Picco. Programming wireless sensor networks: Fundamental concepts and state of the art. *ACM Computing Surveys*, 2011. To appear. Available at: www.sics.se/~luca/papers/mottola10programming.pdf.