
The adoption of precision agriculture has been enabled by the arrival of the new ICT technologies included in the wider context of the Internet of Things. We advocate that formal software engineering models and methodologies can provide support to the design of networks for precision agriculture. We propose here the process algebraic framework as a formal support for understanding the possible data-flows for decision-making, once identified from where data are to be gathered. In particular, we exploit IoT-LySa, a recently introduced process calculus, through which we model a possible agriculture scenario. We consider grape cultivation, with particular attention to a careful usage of water. The static analysis developed for IoT-LySa can be used for predicting the interactions among nodes, how data spread from sensors to the network, and how they are manipulated, so helping in water preservation.

KEYWORDS: Precision Agriculture, Internet of Things, Formal Methods, Process Algebras.

INTRODUCTION

Precision agriculture is a management concept that is based on observing, measuring and responding to variations of micro-climatic parameters, as well as parameters of plants and soil, in order to optimise resources, such as water, fertilisers and plant protection products, so to drive agricultural actions.

As a result, productivity can be improved and the environmental impact minimised, thus supporting sustainable agricultural practices.

One of the enabling ICT technologies in this area is that of Wireless Sensor Networks, included in the wider context of the Internet of Things (IoT). These networks consist of devices (i.e. the “things”) that observe the physical environment, and of processing nodes that cooperatively make decisions and issue operations on the environment, depending on these observations.

We argue that formal software engineering models and tools can provide support to the design of networks for precision agriculture. A formal description of these networks can be indeed useful for allowing designers to focus on the needed requirements in terms of information flow. Once focussed on what is needed, designers can focus on how to obtain it in terms of implementation.

We propose to exploit process algebras as the enabling formal methodology in this framework. A process algebra is an abstract, yet expressive programming language that provides a set of primitives for modelling the behaviour of a system composed by many independent and interacting components. They come endowed with well-founded and established techniques for reasoning on the design of the system and its optimisation. In particular, modelling can help us in understanding the possible data-flows for decision-making, by identifying from where data are to be gathered and which kinds of aggregation functions are really necessary. Furthermore, models offer a way to evaluate how effective is the precision of the network under...
development, allowing designers to understand how to tune data collection in terms of granularity of the observed areas, of frequency of sampling, and so on.

We resort to the process algebra IoT-LySa (Bodei et al., 2017, 2016b) to model and reason about Wireless Sensor Networks for precision agriculture. We illustrate below our proposal, with the help of an example of a possible agriculture scenario of grape cultivation, in which water must be preserved.

**A PRECISE IRRIGATION CONTROL SYSTEM FOR A SMART VINEYARD**

**Smart irrigation**

Smart irrigation control systems are an efficient and sustainable way to manage water, i.e. one of the essential and critical resources in agriculture.

A precise irrigation control system must be able to:

- identify the variability conditions of soil and crop with respect to space and time, and introduce and measure the corresponding quantitative parameters;
- collect and analyse these measurements using an appropriate scale and frequency;
- make the appropriate decisions and plan the consequent operations with the aim of irrigating with the appropriate amount of water each sub-area.

We consider the irrigation system AgriSens, based on a Wireless Sensor Network and developed at IIT Mumbai for monitoring and irrigating grape crops at Sula Vineyard, Nashik (India) (Shah and Da, 2012).

Given the shortage of water in this region, the main task of the system is to regulate irrigation according to evapo-transpiration, a variable parameter that measures water demand of the crop.

This parameter depends on several factors, among which weather, soil moisture, kind of plant and stage of development. Irrigation is needed when evapo-transpiration exceeds the supply of water coming from the soil or from precipitations. AgriSens consists of a combination of wired and wireless sensors that collect soil data like pH, moisture, temperature and so on.

These battery-powered sensors are deployed at a grid of 30 m by 30 m, and can transmit in a range of [30 m, 1000 m]. The collected data are sent in a multi-hop manner to a base station node. The base station node performs a first elaboration of data and then transmits the results via a mobile phone technology to the remote Agri-information server in Mumbai.

![Fig. 1 - Organisation of the irrigation control system.](image-url)
The aggregated data are further processed and stored in the remote server. The irrigation control system decides the suitable irrigating actions for each sub-area in the crop: e.g. if in one of them the level of soil moisture goes down a given threshold, then sprinkler actuators are activated. Users, i.e. farmers and researchers, can directly access the server agri-data at any location and help in decision-making.

Our process algebraic model of the smart vineyard

To give an overall picture of our methodology, we model the AgriSens system in IoT-LySa.

The whole system network, illustrated in Figure 1 and described in Table 1, is a pool of nodes running in parallel that can communicate with each other. Each node, uniquely identified by a label, consists of control processes and, possibly of sensors and actuators. Communication is multi-party: each node can send information to a set of nodes, provided that they are in the same transmission range. Outputs and inputs must match in order to communicate. In more detail, output is modelled as meaning that the tuple of messages \( E_1, \ldots, E_k \) is sent to the nodes with labels in L. Intuitively, a message \( E_i \) contains a value, either a piece of raw data gathered by a sensor or the result of an aggregation of other data, e.g. the average of moistrures collected by several, contiguous devices.

Input is instead modelled as

\[ \text{input} = \text{pattern matching}. \]

In receiving an output tuple of the same number of elements, the communication succeeds, provided that the first \( j \) elements of the output match the corresponding first elements of the input, and then the variables occurring in the input are bound to the corresponding terms in the output, i.e. the values of the output expression are assigned to the input variables.

Each base station node \( N_{bs_{ij}} \) is connected to a bunch of sensors \( S_{bs_{ij1}}, \ldots, S_{bs_{ijk}} \) that sense the environment in the crop sub-area controlled by the base station control process \( P_{bs_{ij}} \) and write the sensed values on its store. The node also includes other components that we omit here because irrelevant. Similarly, the action \( \tau \) denotes internal actions of the sensor we are not interested in.

\[ (1, \text{StartIrrigation}) : \]
\[ (w_{11} < th_{11}) ? \]

\[ (1, \text{StopIrrigation}) \]*

\[ \vdots \]

\[ (n, \text{StartIrrigation}) : \]
\[ (w_{n1} < th_{n1}) ? \]

\[ (1, \text{StopIrrigation}) \]*

User \( t \) with \( t \in [1, h] \) \( N_{ust} = \ell_{ust} : [P_{ust}] \) meaning that the tuple of messages \( E_1, \ldots, E_i \) is sent to the nodes with labels in L.
The node base station \( N_{bs_i} \) collects data \( x^1, \ldots, x^k \) from its sensors, processes them with the help of the filter and aggregation functions \( f_1, \ldots, f_m \), and transmits its results to the Cluster Head node \( N_{ch_j} \). The communication is performed as explained above: e.g. the output performed by the process \( P_{bs}, \) matches the input performed by \( P_{ch} \), and therefore the variable \( x^m \) is bound to \( f(x^1, \ldots, x^k) \), the variable \( x^m \) is bound to \( f(x^1, \ldots, x^k) \), and so on. The construct \( \{ \ldots \} \) implements the iterative behaviour of processes and of sensors.

Each Cluster Head node \( N_{ch_j} \) controls a subset of the base station nodes \( N_{bs_i} \) that send it their data (\( x \) stands for the array \( \{ x^1, \ldots, x^m \} \)). This control is done upon receiving the relevant data, that are then aggregated with the functions \( agg(l), \ldots, agg() \).

The Cluster Head node then sends the obtained results to the Agri-Information server \( N_{as} \).

The Agri-Information server \( N_{as} \) processes the data sent by all the Cluster Head nodes \( N_{ch_1}, \ldots, N_{ch_n} \) and makes its decision on irrigation. If the server detects that some water is needed in the area controlled by the Cluster Head node \( N_{ch_j} \) (i.e. if the water demand \( w_{j1} \) is over a given threshold \( th_{j1} \)), it sends a “start irrigation” order, and conversely, when it detects that there is sufficient water (i.e. if the water demand \( w_{j1} \) is below a given threshold \( th'_{j1} \)), it transmits a “stop irrigation” order. The relevant Cluster Head node \( N_{ch_j} \) transmits the received orders to the corresponding actuators \( A_j \) triggering the irrigation sprinklers. Furthermore, users \( N_{ust} \) can access the data processed and stored by the Agri-Information Server \( N_{as} \), e.g. for checking whether there are the conditions for particular manual processes, like pruning. For the sake of simplicity, we omit the specification of the components of the users’ nodes \( N_{ust} \).

Note that wireless communication depends on the transmission range. For instance, base station nodes and sensors cannot directly communicate with the Agri-Information Server and therefore they have to rely on multi-hop communication via Cluster Head nodes, which instead can communicate, via a mobile phone technology, with the Server in Mumbai.

**DISCUSSION**

In (Bodei et al., 2016a, 2016b, 2017) we equipped IoT-LySa with a static analysis that describes the interactions among nodes, how data spread from sensors to the network, and how data are manipulated. A static analysis is able to safely approximate and predict system behaviour, by only inspecting its specification and without running it, as done in the model checking techniques, which require to consider each reachable state of a system.

The loss in precision is compensated by the gain in efficiency: computing the prediction is far less expensive than computing all the possible dynamic evolutions of the system. The result of the analysis can be exploited as the basis for checking and certifying various properties of precision agriculture design, e.g. reliability of devices and nodes. Applications succeed if the communications among nodes and sensors are reliable. Problems can arise from energy consumption, to which sensors are quite sensitive, and from accidental damages to devices. We can use our analysis in a “what if” fashion, by wondering what happens if a certain node is compromised or damaged: are the required overall functionalities still guaranteed in case of adverse and unstable operating conditions? If the answer is no, the designer should rethink the system. For instance, the designer can add further communications in order to make up for the possibly compromised communications, and consequently improving the accuracy and the reliability of the collected data.

The IoT paradigm introduces large sets of data that need to be processed to make decisions on actuators. Such systems should be resistant against “bad” data that may come from possibly damaged sources (sensors and memory locations), because of accidental failures or security attacks. Making critical decisions, in dependence of data coming from unreliable sensors can have severe consequences in security. To this aim, we exploit the fact that IoT-LySa is an extension of LySa (Bodei et al., 2005, Gao et al., 2008), a calculus designed for studying security, in particular of cryptographic protocols. In the present scenario, we could resort to taint analysis. Taint analysis predicts how information flows from specific data sources to the computations that use these data (Schoepe et al., 2016, Schwartz et al., 2010). The basic idea is to classify unreliable data sources as tainted and the others as untainted, and then mark the derived data as well, through suitable propagation rules. In (Bodei and Galletta, 2017), some of the authors proposed a Control Flow Analysis for statically predicting how tainted data spread across an IoT system. The analysis can be further enriched with causal information, as in (Bodei et al., 2013, 2015), in order to capture the possible causal relations among communications: this could help in reconstructing how data are derived. By exploiting analysis in our agricultural framework, we can determine possible unreliable sensing data, e.g. those coming from sensors that are not reliable because of physical damages or failures. Hence we can address fault-tolerance aspects, by tracing unreliable data and by investigating their impact on the aggregation functions from which depend the most critical decisions.

Furthermore, the proposal in (Bodei and Galletta, 2016) is a preliminary step, still based on IoT-LySa, to infer quantitative measures on systems evolution. The derived quantitative evaluation is exploited to establish the cost of communications, in particular those that can be prone to security attacks, and the possible security countermeasures. Costs can be computed in terms of time overhead, energy consumption, bandwidth, and so on. All these factors must be carefully evaluated for achieving an acceptable balance among usability of the system and the cost required to ensure the desired level of performance and security.

**ACKNOWLEDGMENTS**

Partially supported by Università di Pisa PRA_2016_64 Project Through the fog.
REFERENCES


(ms. pres. 15 gennaio 2017)