A Multi-Agent Architecture for Quantified Fruits: Design and Experience

Jean-Pierre Briot[⋆†] Nathalia Moraes do Nascimento[†] Carlos José Pereira de Lucena[†]

[⋆]Sorbonne Universités, UPMC Univ Paris 06, CNRS
Laboratoire d'Informatique de Paris 6
Paris, France
jean-pierre.briot@lip6.fr

[†]Laboratório de Engenharia de Software
Departamento de Informática, PUC-Rio
Rio de Janeiro, Brazil
nnascimento,lucena@inf.puc-rio.br

Abstract - *The concept of* Quantified Self *is about connected objects* self-monitoring *their human owner (e.g., a watch measuring heart rate, etc.). A natural transposition is in self-monitoring arbitrary things, therefore named* Quantified Things. *In this paper, we present the case of self-monitoring agricultural products. We discuss the rationales for the design of a Quantified Fruit* multi-agent architecture *for* self-monitoring *and* self-prediction *of the* maturation *of fruits. The architecture includes 6 different types of agents, the 2 more specific ones being respectively, the self-controller equipped with various sensors and the self-prediction module. Our current implementation uses an* Arduino microcontroller board *with* 5 sensors *(measuring respectively: temperature, light, humidity, hydrogen and methane). The prediction module uses a* neural network. *We have implemented the architecture and have conducted various* experiments*, storing bananas in diverse settings: room, refrigerator, in a box, with other fruits, etc. The paper discusses the architecture, its current implementation, experiments and current results. Future issues (scalability, collaborative prediction, etc.) are also addressed.*

Keywords – Quantified Self; Quantified Things; Internet of Things; microcontroller; software; architecture; design; implementation; agent; multi-agent system; monitoring; prediction; machine learning; neural network; fruit; banana; maturation; logistics.

1. Introduction

As Swan [1] suggests, the concept of *Quantified Self* (QS) represents the capacity for connected objects to self-measure and self-monitor their human owner. Examples are connected watches or phones that measure heart rate, pressure, exercising habits, etc. Capacities for analysis, patterns detection and prediction (using statistical analysis and machine learning techniques) may be included in order to infer personalized monitoring and diagnostic, e.g., for health monitoring.

In this work, we investigate the adaptation of this idea to arbitrary things, therefore named *Quantified Things* [2]. Some early examples are Quantified Cars [3], using the large electronic monitoring and control facilities of a car to monitor, diagnose and control various features of a car. Swan suggests the use of "QS car chips" to collect cars automotive data and store these informations in a cloud database. By using a mobile application, users could have access to their car's information, such as maintenance records, suggested and scheduled maintenance, and take more accurate action as a result.

We have decided to address the case of agriculture food products which, to our knowledge seems a domain yet little explored. In particular, the lifecycle of fruits has an important impact on its economy. An important issue is indeed to minimize the loss of fruits too mature to be consumed and at the same time to minimize the risk of shortage of products for the consumers. This is specially true in the case of bananas, a fruit having a relatively short ripening period, and very much depending on various conditions (temperature, humidity, light, aeration) [4]. Therefore, important decisions must be taken at various steps of the lifecycle: when to best harvest the fruits, depending of the expected travel (type and duration) to the consumer, how to best transport them, how to store them, at a large scale in a storage or a grocery store, down to the consumer house, etc.

In order to explore these issues, we have designed a prototype multi-agent architecture for Quantified Fruits. Its objective is self-monitoring and self-prediction of fruit maturation. We have tested the architecture in the case of bananas and have evaluated it as a proof of concept. The proposed architecture includes various types of agents, implemented in the JADE framework: an Arduino programmable microcontroller board with various sensors (temperature, light, humidity, hydrogen, methane), a user interface and a neural network-based prediction module. We have selected these sensors based on some works [5, 4] that investigate various factors that interfere on fruit's perishability.

2. Related Work

Some researchers have proposed the use of sensors in the agriculture supply chain. Most of them investigate how technology can be used to improve farming practices, such as collecting information about the land conditions and climate variability and helping farmers to avoid inappropriate farming conditions [6, 7]. However, even though the percentage of agriculture food products losses after picking

DOI reference number: 10.18293/SEKE2016-102

step is extremely high, there have been very few investigations into what are the satisfactory conditions to prolong perishable shelf life in the other supply chain steps, such as distribution. Lang et al. [8], for example, suggest the development of an intelligent container for monitoring temperature parameter during food transportation. Nonetheless, they do not present experimental results. In addition, their model takes only temperature into account for monitoring the quality of the product.

Examples of software architectures for IoT are the FedNet service-based framework [9] and the multi-agent-based middleware ACOSO [10]. As we will see, our architecture is more light weight and more specific to our objective.

3. Application Scenario

3.1. Setting

In our first scenario, we consider a user having at home a set of bananas that he bought and he would like to know what are the best ways to store bananas and how to predict their maturation. We have selected the case of bananas, because: 1) the interaction between the lifecycle of tropical fruits and the logistics for maturing, transporting and storing them is critical and with an economic impact; 2) they have a relatively short lifespan, therefore we can conduct shorter experiments. Different ways of storing bananas may be considered and tested: in a room, in a refrigerator, in a box, alone or with other fruits, etc. A self-monitoring device (described below) is associated to a banana (located on its side) in order to provide the self-monitoring capacity.

3.2. Requirements

Some of the requirements for our application are:

- a software architecture to autonomously provide self measurement and prediction of fruit maturation. It should provide some pro-activeness capacities, such as to self assess its prediction accuracy and if necessary revise its prediction model.
- the architecture must be decentralized, interoperable and light weight, in order to work with different types of microcontrollers, resources and constraint settings (local computing, cloud, network protocol, etc.).
- the architecture should be easy to deploy (with some auto-deployment and configuration ability) and evolvable at design time as well as run time (discovery of new quantified things devices and modules, specially in the context of collaborative applications, see § 6.4).
- a user interface running on a smartphone (or tablet) for the user to control experiments and input some data.

4. Proposed Architecture

4.1. Conceptual Architecture

In order to offer a decentralized and collaborative architecture with such requirements (autonomy, pro-activity, decentralization, inter-operability, easiness to deploy, and furthermore ability to seamlessly incorporate future evolutions, such as collaborative Self Things, see § 6.4), a natural choice was to use a multi-agent approach (see for instance [11] for a more detailed argumentation). Therefore, the various autonomous modules (monitoring, learning module, data base, user interface, etc.) are encapsulated into various interacting agents. The conceptual architecture proposed is shown at Figure 1. It includes the following components:

- `QuantifiedFruitAgent`, which encapsulates the microcontroller and its sensors used for self-monitoring.
- `PredictionAgent`, which encapsulates the machine learning module to make the prediction.
- `TrainingAgent`, which encapsulates the training algorithm which will set up `PredictionAgent` parameters.
- `DataBaseAgent`, encapsulating the various data of the experiments, notably the training set.
- `UserInterfaceAgent`, encapsulating the interface with the user.
- `PlugAndPlayAgent`, responsible for discovering a new QuantifiedFruit, creating a new `QuantifiedFruitAgent`, configuring and connecting it to the architecture components.

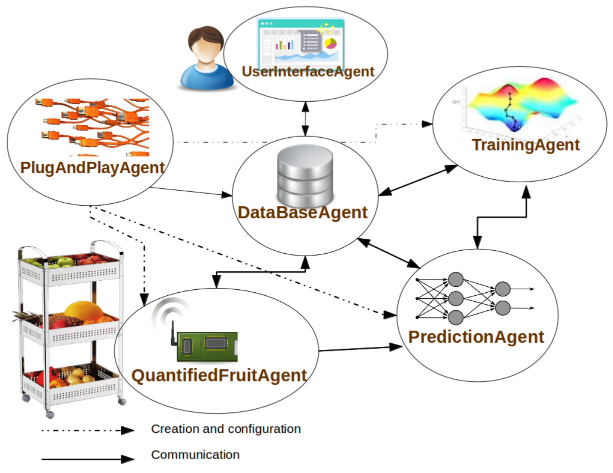

Figure 1: Conceptual architecture.

`PlugAndPlayAgent` creates a `QuantifiedFruitAgent` for each Arduino board that is connected to the system and associates it with a specific `PredictionAgent`.

DatabaseAgent is shared among all agents, allowing them to share information, such as the self-tracking and prediction data. It could be noted that the database does not need to be an agent, but encapsulating it into an agent (agentifying it) brings uniformity to the architecture and the communication between its components, as well a potential of flexibility for the future.

4.2. Current Implementation and Configuration/Deployment

Current implementation and configuration/deployment of the architecture is as follows. Each agent is implemented as a JADE agent, using the JADE multi-agent infrastructure [12]. JADE provides the support for the interoperability and the distribution of the agents.

- QuantifiedFruitAgent encapsulates an Arduino microcontroller [13] and its 5 sensors, respectively measuring: methane, hydrogen, temperature, humidity, and light. It runs on a Java server. There is one QuantifiedFruitAgent for each Arduino board connected to the system.
- PredictionAgent encapsulates an artificial neural network (ANN) used for prediction. It runs on a Java server. There is one PredictionAgent, shared by the different QuantifiedFruitAgent agents.
- TrainingAgent encapsulates both backpropagation and prediction error minimization algorithms for training the neural network [14]. It runs on a Java server.
- DataBaseAgent encapsulates a data base containing the various data of the experiments. It is currently implemented as a simple table. There is one DataBaseAgent shared by all agents.
- UserInterfaceAgent encapsulates the user-interface running on a smartphone or a tablet, implemented in Java. It runs on a smartphone.
- PlugAndPlayAgent is using a discovery protocol similar to Jini Lookup Discovery Service [15]. It runs on a Java server.

In current configuration, there is only one PredictionAgent and one ConfigurationAgent. This means that there is no heterogeneity and all fruits tested are considered to be of the same type. A larger experiment may introduce different types of fruits and associated PredictionAgent agents.

Note that current configuration of the architecture is obviously not scalable, but its configuration can be adapted as will be discussed in § 6.3. More generally speaking, depending on the availability of resources on the underlying computing and communication architecture, we may decide to allocate different agents on different spaces (local, global/shared, hierachical. . .) and also make various decisions on what agents should be shared or replicated.

Prediction Agent

We have decided to use an artificial neural network (ANN) architecture for the prediction module. The reason is as following: ANNs are well known architectures and they have proven their efficiency and moreover versatility. As opposed to linear or polynomial regression modules where one has to *a priori* select a model (linear, quadratic, cubic, including product of features, etc.), the model of a neural network is generic enough although some configuration has to be decided (e.g. the number of hidden layers, the number of units of the hidden layer(s)).

The neural network includes an input layer with 5 units (corresponding to the 5 parameters produced by the 5 sensors), one hidden layer with 4 units and an output layer with one unit (corresponding to the number of days predicted).

The neural network is implemented in Java. We have also implemented a second version in the Octave/Matlab numerical computation programming language, in order to exploit vectorization of data computation and to conduct further analyses (see § 5.4).

Training Agent

Our current method for training the neural network (adjusting the weights of the neuron connexions in order to minimize the error (differences) between predicted and target values) is quite standard: 1) using backpropagation algorithm (to compute the gradients) [14]; 2) combined with an algorithm to minimize the cost function (prediction error) – we have tried out batch gradient descent well as generic optimization algorithms (from off-the-shelf libraries).

Note that, in addition to traditional off-line learning approach, we also experimented with a (simplified) incremental learning approach, where PredictionAgent proactively self-assesses its prediction accuracy and if necessary requests TrainingAgent to incrementally update its prediction model by launching a new learning phase on the new example(s) (in a similar way to on-line learning).

User Interface Agent

It runs on a Java mobile application in order to allow mobile users to monitor fruit storage. This application lists all fruit storages that are connected to the system. After the mobile user selects one of these fruit storage, UserInterfaceAgent will retrieve from the database all the respective information – monitoring data and prediction – and show it on the interface. In addition, this application has an input field for the user to enter the effective (observed) fruit lifespan. UserInterfaceAgent will then communicate this new experimental data to DatabaseAgent.

5. Experiments and Evaluation

5.1. Experimental Setting

The user will try various ways for storing a banana, taking four condition parameters into account: (i) dark (i.e. in a closed or open box); (ii) room (i.e. the box being stored in a fridge or at room temperature); (iii) rotten fruit (i.e. in a box alone or putting together with a rotten); and (iv) ripe fruit (i.e. putting together with a ripe fruit).

Below, we detail four of the possible settings for storing a banana (summarized at Table 1 and depicted at Figure 2):

(a) In an open box, at room temperature, alone;

(b) In an open box, together with a rotten fruit;

(c) In the fridge, with a ripe fruit;

(d) In a closed box, at room temperature, with a rotten fruit.

Table 1: Configuration of experiments at Figure 2.

Experiment	Box		Room		Rotten Fruit		Ripe Fruit	
	open	*close*	*room*	*fridge*	*yes*	*no*	*yes*	*no*
(a)	X		X			X		X
(b)	X		X		X			X
(c)	X			X		X	X	
(d)		X	X		X			X

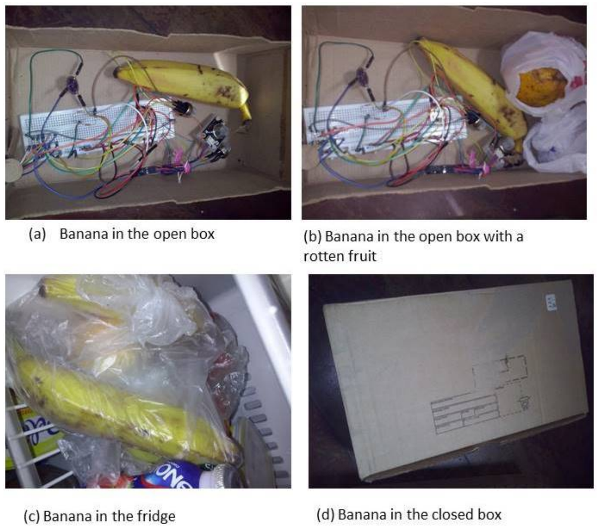
(a) Banana in the open box
(b) Banana in the open box with a rotten fruit
(c) Banana in the fridge
(d) Banana in the closed box

Figure 2: Examples of scenarios.

For each setting, a user creates a new experiment on his smartphone user interface. Then, he triggers the measurement of the parameters (light, temperature, methane, hydrogen and humidity), which will be recorded in the database. He then later checks (usually every day) the maturation of the fruit and reports on the interface when the starting maturation occurs. This process was essential to elaborate an initial database to improve system's predictions. In practice, we have conducted several experiments in parallel, putting a dozen of bananas in different settings and monitoring in parallel their respective maturation process.

Following standard methodology in machine learning, we have partitioned our dataset into a training set and a testing set. (We also have used a cross validation subset for detailed analysis, see § 5.4).

5.2. Training Set

Table 2 shows a subset of the training set used, which represents the data collected from the experiments illustrated at Figure 2. At the beginning of each experiment, `QuantifiedFruitAgent` collects the measured values: temperature (abbreviated Temp.), which is registered in Celsius (C), relative humidity (RH), hydrogen gas (Hyd.), methane gas (Met.), and luminosity (Lum.). Values of gas sensors are recorded according to the sensor output value (V.). At the end of each experiment, the user reports the "actual" fruit lifespan (this information is subjective since in current experiments naked-eye observation determines it).

Table 2: Subset of the training set.

Temp	RH	Hyd	Met	Lum	Lifespan
27.62	70.22	2	184.0	15.0	14
28.02	72.53	8	275.0	10.0	5
27.81	72.75	3.0	258.0	3.0	10

5.3. Test Set

Table 3 shows results for a subset of the test set. This example, which was performed outside the fridge and in an open box, shows a good prediction. The system predicted thirteen days, and the user reported that the banana spoiled in approximately twelve days.

Table 3: Subset of the test set.

Temp	RH	Hyd	Met	Lum	Lifespan	
					Observed	*Predicted*
28.21	70.24	3.0	183.0	16.0	12	13

5.4. Results and Discussion

We believe that these first experiments are promising, the prediction module showing good prediction accuracy. Obviously, we need to conduct more experiments with different settings to collect more data.

We have conducted some analysis of our prediction module. Figure 3 shows the validation curve, which compares the evolutions of the prediction error for the training set (we will name it *training error*, depicted in a blue solid line) and of the prediction error for the cross validation set (*cross validation error*, depicted in a green dashed line) for various (increasing) values of λ (the regularization parameter used to control overfitness). The figure shows that `0.01` is a good value for λ as cross validation error is minimal. For a smaller value, there is some variance (overfitness) because the training error is almost null and the cross validation error is significant, showing the poor generalization of the model. For a larger value, the cross validation error is increasing (note that the training error is also increasing), showing an increasing bias (underfitness).

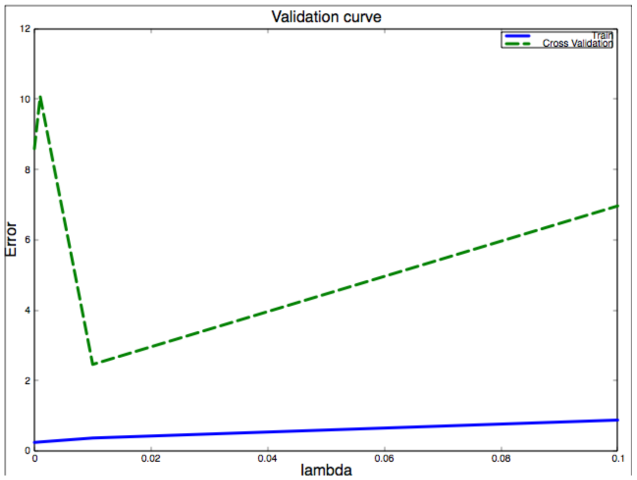

Figure 3: Validation curve.

Figure 4 shows the learning curve, i.e., the evolution of prediction error depending on the size of the training set. The figure shows that the training error is almost null and that the cross validation error stays low, confirming that the model has low bias and low variance. These preliminarily analyses are encouraging. We are conducting more experiments in order to collect more data in order to further improve the model.

6. Open Challenges

6.1. Learning Algorithms

The preliminary tests that we have conducted show that the neural network has a good prediction accuracy. But we need to conduct more experiments in order to construct a

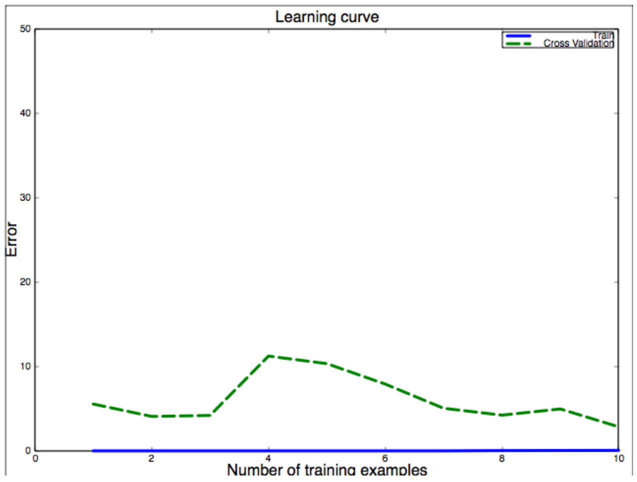

Figure 4: Learning curve.

sufficient and representative data set. We will also continue to conduct analysis of the behavior of the neural network. Note that our architecture is generic and we may consider other types of `PredictionAgent`, with alternative prediction modules and learning algorithms.

6.2. Genericity

Our architecture is actually generic and may be used for other purposes. Current implementation served as a proof of concept of the architecture. But current implementation provides a unique type of prediction and configuration strategy. It is actually easy to introduce heterogeneity and various kinds of quantified things, prediction and configuration strategies (and associated agents).

Another dimension of genericity is to use the neural network as a controller (and not for prediction). Actually, we have instantiated a preliminary version of the architecture [2] which since then has been completely redesigned, and tested it on a scenario of simulated traffic management, where controllers control semaphores at the crossing of roads. The behavior of the semaphores is evolved using evolutionary algorithms, inspired by evolvable architectures for robotics. This work has been described in [16].

Last, we are planning to reuse and experiment our architecture on other types of applications, such as for instance distributed self monitoring of air pollution in a city through mobile users (e.g., on bicycles and carrying the sensors/microcontroller device).

6.3. Scalability

Our current implementation is operational but is not scalable. Meanwhile, we may relatively easily reconfigurate and deploy the architecture in a more distributed setting. For example, `PlugAndPlayAgent` could be evolved into a hierarchical architecture mapped for network subdomains.

In addition, `QuantifiedFruitAgent` could directly run on the microcontroller[1].

6.4. Collaboration

A future obvious direction is in making similar quantified things collaborative. In the case of bananas or fruits, various quantified things could exchange and share (and integrate) various experiences and data in order to extend and refine the analysis and predictive abilities. We can imagine scenarios for the fruit farms (plantation and conditioning), storage and delivery facilities.

Note that, in addition of offering predictions about fruit lifespan, this system could be adapted to provide other kinds of predictions, such as the percentage of fruit production that could be lost under specific transportation conditions.

6.5. From Prediction to Decision

Besides making predictions, this tool could also be extended to make suggestions and act on its own. For example, the device could make suggestions for temperature changes in real time. If we added a cooler device to current system, it could autonomously adjust temperature. Note that, as was explained in § 6.2, a preliminary version of our architecture has already also be used for control [16].

7. Conclusion

In this paper we have described a multi-agent architecture of quantified fruits for self-predicting maturation of fruits. It includes 6 types of agents, among them: a self-controller equipped with various sensors measuring storage conditions (light, temperature, humidity, etc.) and a self-prediction module based on a neural network. Our current implementation uses an Arduino microcontroller board with 5 sensors. We have implemented the architecture and have conducted various experiments with real settings and real data (storing bananas in diverse settings: room, refrigerator, in a box, with other fruits, etc.).

We believe these preliminary results to be promising. We think that they open the way for more experiments in testing the architecture in a more distributed and collaborative settings (various fruits and various stages of storing). We hope this tool may lead to improvements both in transportation methods by distributors, consumers and retailers, as well as their storage patterns and practices (refrigeration, packaging, etc.). Last, we believe this prototype architecture could be adjusted for other types of applications, such as collaborative monitoring of city air pollution (e.g., by citizens riding bicycles equipped with such architectures).

[1]Nonetheless, JADE can only be used to implement agents to execute in Java-compatible systems. Thus, a JADE agent cannot (yet) be directly deployed on an Arduino board since it currently only provides support for development of C programs. Therefore, we have created a software layer to interface JADE with Arduino boards via sockets.

Acknowledgements to CAPES, CNPq and FAPERJ for their support through scholarships and fellowships.

References

[1] M. Swan, "Sensor mania! The Internet of Things, wearable computing, objective metrics, and the Quantified Self 2.0," *Journal of Sensor and Actuator Networks*, vol. 1, no. 3, pp. 217–253, 2012.

[2] N. M. Nascimento, C. J. Lucena, and H. Fuks, "Modeling quantified things using a multi-agent system," in *IEEE / WIC / ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*. IEEE, 2015, pp. 26–32.

[3] M. Swan, "Connected car: Quantified self becomes quantified car," *Journal of Sensor and Actuator Networks*, vol. 4, no. 1, pp. 2–29, 2015.

[4] D. Johnson, N. Hipps, and S. Hails, "Helping consumers reduce fruit and vegetable waste: Final report," Waste and Resources Action Programme (WRAP), U.K., Tech. Rep., 2008.

[5] A. Boe and D. Salunkhe, "Ripening tomatoes: Ethylene, oxygen, and light treatments," *Economic Botany*, vol. 21, no. 4, pp. 312–319, 1967.

[6] R. Khan, S. U. Khan, R. Zaheer, and S. Khan, "Future internet: The Internet of Things architecture, possible applications and key challenges," in *Frontiers of Information Technology (FIT), 2012 10th International Conference on*. IEEE, 2012, pp. 257–260.

[7] J. Burrell, T. Brooke, and R. Beckwith, "Vineyard computing: Sensor networks in agricultural production," *Pervasive Computing, IEEE*, vol. 3, no. 1, pp. 38–45, 2004.

[8] W. Lang, R. Jedermann, D. Mrugala, A. Jabbari, B. Krieg-Bruuckner, and K. Schill, "The "Intelligent Container" - A cognitive sensor network for transport management," *Sensors Journal, IEEE*, vol. 11, no. 3, pp. 688–698, 2011.

[9] F. Kawsar, T. Nakajima, J. H. Park, and S.-S. Yeo, "Design and implementation of a framework for building distributed smart object systems," *The Journal of Supercomputing*, vol. 54, no. 1, pp. 4–28, October 2010.

[10] G. Fortino, A. Guerrieri, W. Russo, and C. Savaglio, "Integration of agent-based and Cloud Computing for the smart objects-oriented IoT," in *IEEE 18th International Conference on Computer Supported Cooperative Work in Design (CSCWD'2014)*, May 2014, pp. 493–498.

[11] ——, "Middlewares for smart objects and smart environments: Overview and comparison," in *Internet of Things Based on Smart Objects – Technology, Middleware and Applications*. Springer, 2014, pp. 1–27.

[12] F. Bellifemine, F. Bergenti, G. Caire, and A. Poggi, *Jade – A Java Agent Development Framework*. Springer, 2005, pp. 125–147.

[13] Arduino, "Arduino," http://www.arduino.cc/.

[14] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[15] J. Newmarch, *Foundations of Jini 2 Programming*. Apress, 2007.

[16] N. M. Nascimento, "FIoT: An agent-based framework for self-adaptive and self-organizing internet of things applications," Master's thesis, PUC-Rio, Rio de Janeiro, Brazil, August 2015.