PUC

ISSN 0103-9741

Monografias em Ciência da Computação
n° 04/2015

FIoT: An Agent-Based Framework for Self-Adaptive and Self-Organizing Internet of Things Applications

Nathalia Moraes do Nascimento
Carlos José Pereira de Lucena

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22453-900

RIO DE JANEIRO - BRASIL

Monografias em Ciência da Computação, No. 04/2015 ISSN: 0103-9741
Editor: Prof. Carlos José Pereira de Lucena November, 2015

FIoT: An Agent-Based Framework for Self-Adaptive and Self-Organizing Internet of Things Applications

Nathalia Moraes do Nascimento, Carlos José Pereira de Lucena

nnascimento@inf.puc-rio.br, lucena@inf.puc-rio.br

Abstract. The agreed fact about the Internet of Things (IoT) is that, within the coming years, billions of resources, such as cars, clothes and foods will be connected to the Internet. However, several challenging issues need to be addressed before the IoT vision becomes a reality. Some open problems are related to the need of building self-organizing and self-adaptive IoT systems. To create IoT applications with these features, this work presents a Framework for Internet of Things (FIoT). Our approach is based on concepts from Multi-Agent Systems (MAS) and Machine Learning Techniques, such as a neural network and evolutionary algorithms. To illustrate the use of FIoT, we derived two different instances from IoT applications: (i) Quantified Things and (ii) Smart Cities. We show how flexible points of our framework are instantiated to generate an application.

Keywords: Internet of Things, Multi-Agent System, Self-Organizing, Self-Adaptive, Quantified Things

Resumo. A ideia principal da Internet das Coisas (IoT) é conectar bilhões de coisas à Internet nos próximos anos, a exemplo de carros, roupas e comidas. Entretanto, muitos problemas precisam ser resolvidos antes que essa ideia possa ser concretizada. Alguns desses problemas estão relacionados à necessidade de construir sistemas para IoT que sejam auto-organizáveis e autoadaptativos. Este trabalho, portanto, apresenta a elaboração do Framework para Internet das Coisas (FIoT), um Framework para suporte ao desenvolvimento de aplicações para IoT com essas características. Ele é baseado nos paradigmas de Sistemas Multiagente (SMA) e algumas técnicas abordadas em Aprendizado de Máquina, a exemplo de redes neurais e algoritmos evolutivos. Para demonstrar o uso do FIoT, dois grupos de problemas em IoT serão instanciados: (i) Cidades Inteligentes e (ii) Quantificação de Coisas.

Palavras-chave: Internet das Coisas, Sistema Multiagente, Auto-Organização, Autoadaptação, "Quantified Things"

In charge of publications:
Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC-Rio Departamento de Informática
Rua Marquês de São Vicente, 225 - Gávea
22453-900 Rio de Janeiro RJ Brasil
Tel. +55 21 3114-1516 Fax: +55 21 3114-1530
E-mail: bib-di@inf.puc-rio.br
Web site: http://bib-di.inf.puc-rio.br/techreports/

1 Introduction

Internet of Things (IoT) is a broad concept. In general, IoT refers to a global infrastructure of networked physical things interconnected through Internet [1, 2]. The central perspective is that, within the coming years, billions of resources, such as cars, lamps, foods and factory machinery will be connected to the Internet and share information about themselves and their environments. IoT will make it possible to develop a variety of application scenarios, such as smart homes and cities, e-health, environmental monitoring and many others. Smart traffic management is an example of a smart city application, which aims at providing intelligent transportation through real-time traffic information and path optimization [3].

According to the authors in [4], the potentialities offered by IoT make it possible to develop a huge number of applications, a very small part of which is currently available to our society. Most IoT applications are not developed yet because they require scalability beyond millions of devices where centralized solutions could exceed their boundaries. Further, they cannot have fixed deployments or fixed systems configurations, as the environment is in continuous transition [5]. For example, an autonomous application for traffic management depends on the abilities of the traffic light controllers to adapt to changing traffic situations [6]. We can observe that traffic changes according to different time-scales. As the authors describe in [6], a typical workday can be divided into several periods of different traffic situations, including two peak periods with high demands due to commuter traffic.

The truth is that several challenging issues still need to be addressed before the IoT vision becomes a reality [7, 8, 4]. A possible further complication is the fact that the vast majority of present research in universities and industry is paying more attention to operational technology, in order to solve problems related to limited Internet traffic capacity, communication protocols, and network architecture [9]. For example, the authors in [3] discuss the open challenges and future directions in the Internet of Things. They present global addressing schemes, cloud storage, and wireless power as the key elements of the current IoT research. In their opinion, self-adaptive system of systems is an example of the key application outcomes that are only expected in the next decade.

In [5], one of the few works that bring a non-operational IoT approach, the authors relate some open issues which are needed to build elements capable of copping with the changing environments and taking appropriate decisions autonomously. Therefore, they discuss about the importance of creating autonomous and adaptive IoT systems. In an effort to call attention to these issues, a new terminology associated with IoT is emerging: Smart Objects (SOs) or Smart Things. They represent loosely coupled and decentralized systems of cooperating objects. Editors in [5] discuss smart objects and define them as an autonomous, physical digital object augmented with sensing/actuating, processing, interpretation, storing, and networking capabilities.

IoT is a new and exciting approach, and will soon be adopted by the market [3]. To define new frameworks/middlewares [10, 11] for the rapid prototyping, there is a need to facilitate the development process of SOs. Frameworks are general software systems (i.e. systems that consist of abstract and concrete classes), which can be adapted or extended to create more specific applications. According to Ian Sommerville [11], "the sub-system is implemented by adding components to fill in parts of the design and by instantiating the abstract classes in the framework."

Meanwhile, a few framework/middleware approaches have been proposed to support the creation of a SO-based IoT infrastructure [12]. The authors in [13] analyze the existing approaches and discuss their limit in the management of a vast number of cooperative SOs - none of them presented the design of any complex application scenario (i.e. using a vast number of cooperative things, such as traffic scenarios). For example, the authors in [14] developed a middleware for smart objects and affirm that the support of distributed computing entities is the key and novel feature of their approach. Nonetheless, to illustrate the use of their architecture, they present a simple case study, which refers to a smart office environment constituted by only two cooperating things. These works do not show efficiency in complex scenarios, where things must cope with a changing environment and where a sophisticated organization system is required.

According to the authors in [13], to develop SO-based IoT systems, novel software engineering methodologies for dynamic systems need to be defined. Due to proliferation of the Internet and the spread of mobile devices, an increasing number of applications are suitable for using self-organization to fulfill their goals, such as manufacturing control and traffic management [15]. Accordingly, we aim at facilitating the development of smart systems within the Internet of Things domain, providing them with more autonomy, self-adaptive and self-organizing properties. In order to do this, we first propose a model to create smart things. Our approach is based on Multi-Agent Systems (MAS) and adaptive techniques that have been commonly used in robotics to develop autonomous embodied agents (as robots) (see [16, 17]). Thus, we present a framework, named "Framework for Internet of Things"(FIoT), as an effort to facilitate the process of creation of smart things.

The objective of FIoT is to facilitate the creation of diverse applications, such as controllers for car traffic, factory machines, public lighting, and smart homes. Hence, the framework allows the creation of autonomous controllers for groups of homogeneous things which can be connected to the Internet. To illustrate the use of FIoT, we will present examples from two of the Internet of Things applications: (i) Quantified Things and (ii) Smart Cities.

1.1 In what follows we try to addres the following question: Why use MAS and Adaptive Techniques from the Robotic area?

Multiagent Systems (MAS) are widely used to model real-world and social systems, whereas agents are proper to model real entities [18]. MAS provides a useful paradigm for managing large and distributed information handling systems [19]. In addition, an agent could have some characteristics, such as autonomy and social ability, which make MAS suitable for systems requiring self-organization (SO).

Multi-agent systems can gain in robustness, management, and simplicity if they are developed according to the principles of self-organization. However, the models used in Self-Organized Multiagent Systems (SO MAS) tend to be very complex [19]. The use of learning and evolution strategies in a design of SO MAS reveals new possibilities to reduce this complexity [20, 18].

Robotic researchers already have devoted considerable time to study autonomous self-organizing physical systems and their problems [21, 22, 23]. They intend to develop methods that allow robots to learn how to perform complex tasks automatically. A primary focus of contemporary autonomous robotics research is to develop machine learning methods for use in robotic systems. In recent years, a new machine learning method appeared and gained academic and industrial attention: the Evolutionary Robotic (ER) [24, 16, 17]. The primary goal of ER is to develop methods for automatically synthesizing intelligent autonomous robot systems. Evolutionary Robotic have the potential to lead to the development of robots that can adapt to uncharacterized environments, which may be able to perform tasks that human designers do not thoroughly understand [25].

The paper is organized as follows. Section 2 provides a literature survey as related work. Section 3 describes our model and the FIoT framework. Section 4 discusses how the proposed framework can be used to create IoT instances, presenting experiments setup. The paper ends with conclusive remarks in Section 5.

2 Related Work

We first present an overview of frameworks in the literature while addressing the idea of mixing Artificial Intelligence and Internet of Things concepts, especially those with focus on multi-agent systems. We also describe a framework for physical agents that not have focus on IoT, but which provides physical systems with self-organizing and self-adaptive properties.

We know of few research efforts in the literature about smart objects for Internet of Things applications. The authors in [13] aim at providing a clear picture of the suitability of middlewares to support the development of Smart Objects-based Internet of Things systems. The main SO middlewares have been described and compared in their paper based on a set of requirements for smart environments and objects.

Finally such authors listed four middlewares which provide efficient management to develop and deploy SOs: ACOSO (Agent-based Cooperating Smart Objects) [12, 14], FedNet [26], Ubicomp [27] and Smart Products [28]. They make use of different architectural models: the ACOSO is agent-oriented and event-driven, the FedNet is service-oriented, while Ubicomp and Smart Products are component-based.

However, authors in [13] also discuss the limit of these middlewares in the management of a vast number of cooperative SOs, since none of them presented a case study to demonstrate its efficiency in wide scenarios. According to the authors, to develop SO-based IoT systems, novel software engineering methodologies for extreme-scale dynamic systems need to be defined. It is also necessary to include specific abstractions able to deal with system/component evolution that is a typical property of SO systems. The authors argue that agent-oriented methodologies could be exploited by engineers as the basis for formalizing such an effective development method for SOs. Multiagent Systems were widely employed to cope with the main requirements for IoT systems: interoperability, abstraction, collective intelligence and experience-based learning.

The work performed in [9] also proposes a framework for the IoT based on a multiagent System Paradigm. In this sense, the authors listed some requirements needed for developing IoT applications. This list gives support to the domain analysis of our proposed framework. According to authors, requirements are the acquisition of measurements and data from devices, its processing and translation of a context of useful information, and actuation over the environment. Moreover, the approach showed that agents have characteristics that are suitable for those requirements, such as perception, autonomy and social ability.

Despite the fact that the paper presents a real motivation to our approach, it only offers a brief description of a framework components. The authors have also mentioned that there is still the need for detailing every component and give them the intelligent characteristics. Our approach provides intelligence components to develop IoT applications through adaptation and organization algorithms, since we agree that it is crucial to model this type of application.

A framework developed by the Italian Institute of Cognitive Science and Technologies [29], and used to support FIoT is the Framework for Autonomous Robotics Simulation and Analysis (FARSA) [30]. This framework was created to assist research in the area of embodied cognition, adaptive behaviour, language and action. A set of works on Evolutionary Robotics [16, 31, 32, 33] were developed using FARSA or related software. Most of these experiments presents a group of embodied agents that evolves for the ability to solve a collective problem.

Another related work to our approach is the framework presented in [34], Framework for Evolutionary Design (FREVO). The authors in [34] presents Frevo as a multi-agent tool for evolving and evaluating self-organizing simulated systems. The authors affirm that Frevo allows a framework user to select a target problem evaluation, controller representation and an optimization method. However, it concentrates only on evolutionary methods for agent controllers. As a result, this tool can only provide offline adaptations and evolve only simulated environments. In addition, it is often applied in the creation of autonomous robots.

Unfortunately, we can not reuse these referred platforms to control smart objects since it they are very oriented towards the simulation of robotic agents. Furthermore, these platforms have limited communication structure since they do not give support to heterogeneous platforms required by current networks, such as desktop, web, mobile and microcontroller boards.

3 FIoT: Framework for Internet of Things

In this section, we first perform a survey of Internet of Things requirements taking previously published and personal experiences into account. Then we describe our proposed agent-based model to create IoT systems and we show how this model meets these requirements. Our proposed model consists of three layers: (i) physical, (ii) communication, and (iii) application. To facilitate the development process of the communication and application layers of an IoT system, we developed the Framework for Internet of Things (FIoT). Therefore, we also present the FIoT in this section.

During framework developments, three stages must be considered: (i) domain analysis, (ii) framework design, and (iii) framework instantiation [35]. A domain analysis stage provides a survey of domain requirements. In the framework design stage, we used the Unified Modeling Language (UML) diagrams [10] to specify FIoT structure, behavior, and architecture. UML use case [36] and UML activity diagrams [37] are used to assist the description of the main idea of FIoT. In addition, we present the FIoT UML class diagram [35], followed by the analysis of its kernel ("frozen-spots") and flexible points ("hot-spots"). "Frozen-spots"are immutable and must be part of each framework instance. "Hot-spots"represent the flexible points of a system, which must be customized in order to generate a specific application [35].

According to the authors in [35], the abusive use of hot spots in a framework design will inevitably lead to complex software systems. Therefore, the framework designer has to choose the hot spots carefully, neither exaggerating nor creating a far too generic framework.

The instantiation stage is presented in the section 4, performing the generation of new instances through implementation of the FIoT's hot spots.

3.1 Domain Analysis

As we emphasized in the section 1 and 2, we used the works in [5] and in [9] as basis of our domain analysis. We also consider the requirements for the development of self-organizing and self-adaptive applications proposed by the authors in [38].

From an engineering perspective, IoT systems are distributed systems consisting of components (things) that may be physical devices, animals or people. As we aim at giving support to the development of smart things, these components have to autonomously collect data about themselves and their environments and take actions [39]. A smart IoT system can make decisions based on collected data and use dynamic reconfiguration to improve its performance.

All IoT applications share common features, such as to connect and collect data; but they have different features that vary according to specific application scenarios. To assist the development of self-organizing and self-adaptive IoT applications, we performed a discovery domain requirement (R), as follows:

- R1. Design-time description (problem domain):
  - R1.1 To analyze environmental conditions that are associated with the problem goal (e.g., temperature, gases)
  - R1.2 To define how to collect environmental conditions (e.g., a microcontroller board and sensors)
- R2. Decentralization and Interoperability.
- R3. Autonomous things:
  - R3.1 Things should be capable of autonomously sensing/monitoring themselves and their environments
  - R3.2 Actuation over the environment
- R4. Self-adapting capability:
  - R4.1 The individual components or the whole system should be capable of identifying any new condition, failure, or problem by themselves/itself
  - R4.2 Run-time capability of reasoning and of acting/adapting
- R5. To design a software to allow the system:
  - R5.1 To recognize devices in the environment;
  - R5.2 To acquire the data from devices that are collecting environmental data;
  - R5.3 To interface with device sensors; and
  - R5.4 To process and translate the data to a context of useful information.

In the next subsections, we show how our proposed model and framework meet the requirements listed above.

3.2 Agent-Based Model

We developed an agent-based model to be used as a basis for generating different kinds of applications for Internet of Things. Our approach is completely based on Artificial Intelligence paradigms, such as multi-agent systems, neural networks and evolutionary algorithms. We aim at providing mechanisms to automatically recognize and manage things in the environment.

As depicted in Figure 1, our model consists of the design of three layers: (L1) physical, (L2) communication, and (L3) application. Each device in the environment (physical layer) can be recognized and controlled by agents in the application layer.

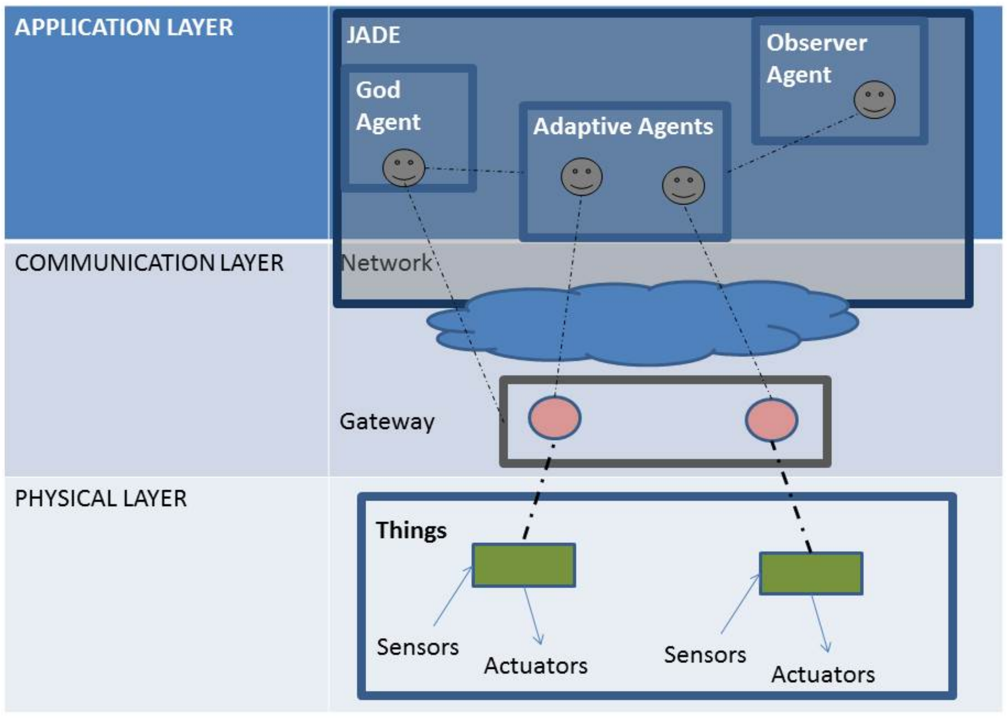


Figura 1: An agent-based model to generate IoT applications.

The physical layer can consist of simulated or real devices (also named as things/objects) and environments. In order to model the physical layer, the project designer has to define the features of things as well as the features of the environment in which these things are situated. He must raise the environmental conditions that need to be monitored (e.g. temperature, relative humidity and car flow). Then he can make specifications for devices. For example, to set their sensors and actuators (i.e. the necessary technology to collect data or make changes on the environment).

The communication layer defines that communication among agents on the application layer and devices must occur using the Internet. Each thing has one address, so an agent can access this address to get and set the necessary information to control the device. We suggest the Java Agent Development Framework (JADE) [40] [41] to implement the communication among agents and things. JADE implements the Foundation for Intelligent Physical Agents (FIPA) protocol for agent communication. It allows the development of an interoperability communication structure, which agents can execute on different platforms and exchange information.

The application layer uses a Multi-Agent System (MAS) to provide services, such as collecting, analyzing and transmitting data from several sensors to the Internet and back again. We aim at providing decentralization, autonomy and self-organizing features to applications through MAS. In addition, we provide the capacity of creating physical agents capable of interacting dynamically with complex environments by using approaches commonly applied in Robotics.

We suggest developing controllers at the application layer to allow autonomous management of devices in the physical layer.

3.3 Central Idea for the Framework Design

A FIoT application consists of the development of three kinds of agents: (i) God Agents [42]; (ii) Adaptive Agents; and (iii) Observer Agents [43]. The primary role of the God Agent is to detect new Things, that are trying to connect to the system. Thus, the God Agent allows the automatic connection of new devices to the system, as a “plug and play”connection. The “plug”in this example, means that the device needs to send a message to the God Agent’s IP address, excluding manual settings. For each connected device, the God Agent creates an Adaptive Agent to control it. An Adaptive Agent is an agent embodied in a Thing, according to the description provided in [44] . While a device represents its body, a JADE software agent contains its controller. The God Agent sets the controller (i.e. the “brain”of the agent) for an Adaptive Agent according to the type of its device (e.g. the number of sensors and actuators). Therefore, the controller creation is a flexible point on FIoT system implementation.

Authors in [45] developed a framework to implement self-adaptive software agents based on the autonomic computing principles proposed by IBM [46, 47]. In order to create adaptive agents, they provided a control loop composed of four activities: collect, analyze, plan and execute. What follows is a brief description of each of these four activities:

- **Collect:** to collect application data;
- **Analyze:** to analyze those data by trying to detect problems;
- **Plan:** to decide what should be done in the case of problems; and
- **Execute:** to change the application due to executed actions.

We customized the control loop used in their work to define the behaviors of the FIoT’s Adaptive Agents. Instead of executing the analyze and plan activities, the FIoT’s Adaptive Agents make decisions based on a controller, which can be, for example, a finite state machine (FSM) or an artificial neural network (ANN), as shown in Figure 2.

Therefore, our Adaptive Agent must execute a sequence of three key activities: (i) collect data from the thing; (ii) make decisions; and (iii) take actions. The task of data

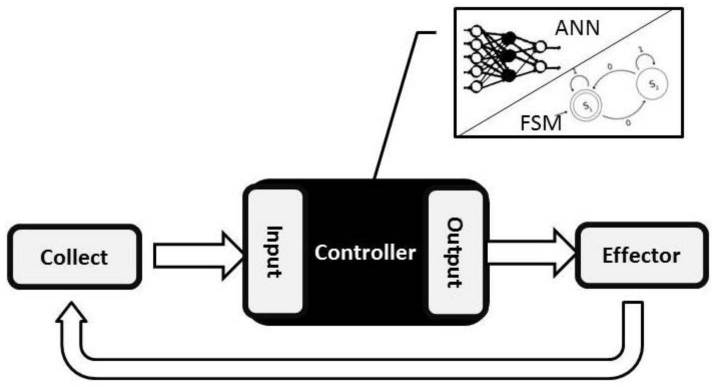


Figura 2: Control loop provided by the FIoT framework.

collection focuses on processing information coming from devices, such as reading data from input sensors. These collected data are used to set the inputs of the agent's controller. Then, the controller processes a decision to be taken by the agent.

Adaptive Agents act according to the controller output. An action (effector activity) can be to interact with other agents, to send messages, or to set actuators data of devices, allowing them to make changes to the environment.

The Observer Agent aims at allowing Adaptive Agents to cope with the changing (dynamic) environments, and at making them capable of adapting to the unexpected. By using an adaptive approach, we expect the emergence of features that have not been defined at design-time, including a sophisticated self-organizing system.

Some researchers [17, 24, 48, 23, 21, 49] investigate the emergence of cooperative or competitive self-organizing multi-agent systems. One of the specifications to generate a cooperative self-organizing multi-agent system is to conduct the adaptive process according to collective evaluations. Self-organizing systems have global goals. Thus, we aim at investigating during the adaption process if a collection of agents are attending together the global goal or not. If the system needs to adapt, the adaptation is performed for the whole multi-agent system. If we conduct the adaptive process according to individual evaluations, the agents may compete with each other. This is the main reason that we provide an Observer Agent to evaluate the global behavior of the collection of Adaptive Agents and to conduct the adaption process of the whole system. Therefore, its main goal is to verify if the Adaptive Agents need to adapt or not. When the actions of agents are far from what it expects, the Observer Agent executes a supervised or unsupervised learning method, such as backpropagation or genetic algorithm.

The process of adaptation consists of generating new configurations for Adaptive Agents' controller and test how agents will behave in the environment. The Observer Agent selects the configuration used when the collection of Adaptive Agents shows a desired global action to set their controller. While the Observer Agent looks for the new controller configuration, Adaptive Agents continue their execution normally.

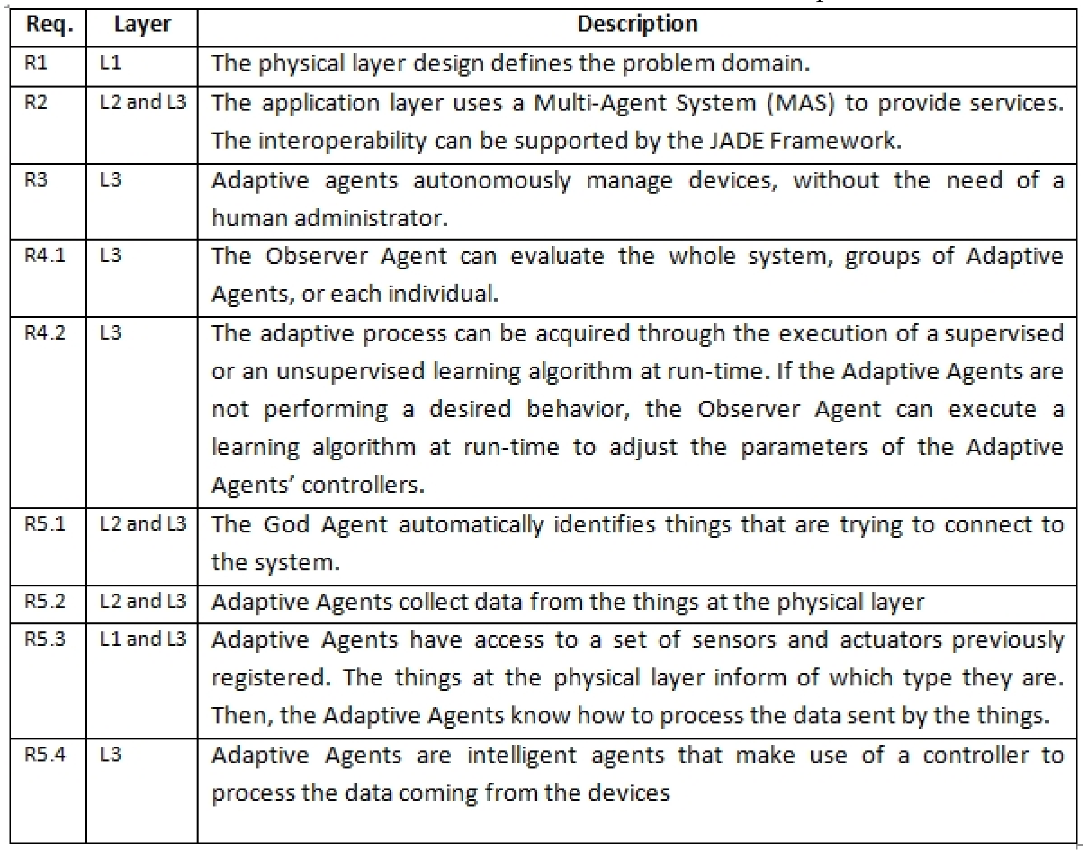
The Observer Agent is tightly coupled to the application being developed. The evaluation process has to be implemented according to the expected global solution. Another variable activity is the generation of new configurations for controllers. It depends on the applied adaptive technique.

As agents execute specific activities to virtualize things and communicate with them at the physical layer, these things must execute the following sequential activities:

- Connect to the Internet
- Send message to the GodAgent, reporting your type of controller. The GodAgent has some controllers already registered. Thus, the type of controller indicates the characteristics of a device, such as the list of sensors and actuators.
- Wait message from GodAgent containing the address of its AdaptiveAgent. Then, the thing will use this address to send and receive the next messages in a cycle:
  - Send message with data sensors
  - Wait message with data to set its actuators.

Table 1 summarizes the model and framework description in this section, and presents how them meet the requirements listed in Section 3.1, according to their layers. FIoT meets the requirements associated with the layers of communication (L2) and application (L3).

Tabela 1: How the model and FIoT meet the IoT requirements.



Req.	Layer	Description
R1	L1	The physical layer design defines the problem domain.
R2	L2 and L3	The application layer uses a Multi-Agent System (MAS) to provide services. The interoperability can be supported by the JADE Framework.
R3	L3	Adaptive agents autonomously manage devices, without the need of a human administrator.
R4.1	L3	The Observer Agent can evaluate the whole system, groups of Adaptive Agents, or each individual.
R4.2	L3	The adaptive process can be acquired through the execution of a supervised or an unsupervised learning algorithm at run-time. If the Adaptive Agents are not performing a desired behavior, the Observer Agent can execute a learning algorithm at run-time to adjust the parameters of the Adaptive Agents' controllers.
R5.1	L2 and L3	The God Agent automatically identifies things that are trying to connect to the system.
R5.2	L2 and L3	Adaptive Agents collect data from the things at the physical layer
R5.3	L1 and L3	Adaptive Agents have access to a set of sensors and actuators previously registered. The things at the physical layer inform of which type they are. Then, the Adaptive Agents know how to process the data sent by the things.
R5.4	L3	Adaptive Agents are intelligent agents that make use of a controller to process the data coming from the devices

3.4 Details of FIoT

As presented in the section 3.2, our model proposes the use of JADE to support the communication among agents and things. FIoT extends JADE, a Java framework to implement multi-agent systems. The project consists of the development of JADE agents, the behaviors of agents, the controller to be used by Adaptive Agents, and the adaptive process to be executed by the Observer Agent. In addition, the system gives support to different interface communication message systems, such as sockets and ACL. We will present the main FIoT classes [11] of the main packages.

The class diagram depicted in Figure 3 illustrates the FIoT classes associated with the creation of agents and their execution loops. As described before, the FIoT agents are represented by the GodAgent, ObserverAgent and AdaptiveAgent classes, which extend the FIoTAgent class. Then, they can access and make changes to the list of controllers (ControllerList class). This list stores all controllers already created by the GodAgent for each thing type (e.g. chair with one temperature sensor, lamp with one presence sensor and one alarm actuator).

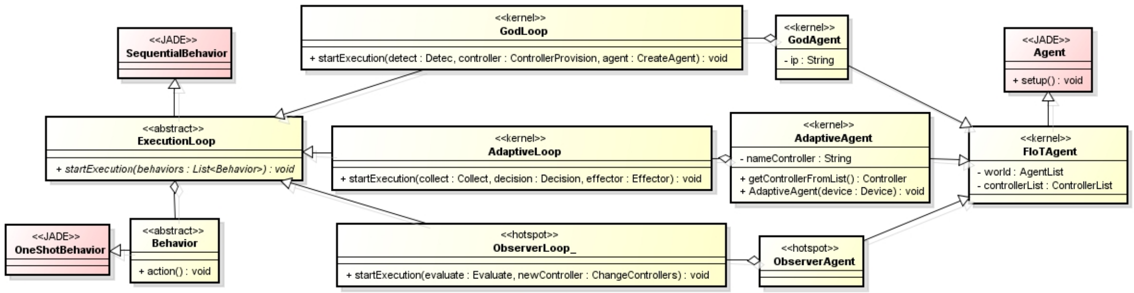

<<JADE>>
SequentialBehavior
<<kernel>>
GodLoop
+ startExecution(detect : Detec, controller : ControllerProvision, agent : CreateAgent) : void
<<kernel>>
GodAgent
- ip : String
<<JADE>>
Agent
+ setup() : void
<<abstract>>
ExecutionLoop
+ startExecution(behaviors : List<Behavior>) : void
<<kernel>>
AdaptiveLoop
+ startExecution(collect : Collect, decision : Decision, effector : Effector) : void
<<kernel>>
AdaptiveAgent
- nameController : String
+ getControllerFromList() : Controller
+ AdaptiveAgent(device : Device) : void
<<kernel>>
FIoTAgent
- world : AgentList
- controllerList : ControllerList
<<JADE>>
OneShotBehavior
<<abstract>>
Behavior
+ action() : void
<<hotspot>>
ObserverLoop_
+ startExecution(evaluate : Evaluate, newController : ChangeControllers) : void
<<hotspot>>
ObserverAgent

Figura 3: Class diagram of FIoT - Agents.

All agents execute sequential behaviors, named as ExecutionLoop: GodLoop, AdaptiveLoop and ObserverLoop classes. The sequential behavior is a type of JADE behavior that provides support to the implementation of composed activities [40]. Thus, the ExecutionLoop is a sequence of smaller actions. For example, for Adaptive Agents, these execution loops are composited of collect, decision and effector activities.

The class diagram depicted in Figure 4 illustrates the collection of behaviors already developed. Activities such as making evaluation and controller adaptation are examples of hot spots. Therefore, new strategies for evaluation and adaptation can be developed to be used by agents. The God Agent's execution loop performs three behaviors: "Detect", "CreateAgent", and "ControllerProvision".

While ObserverAgent access the ControllerList to adapt controllers configuration through ChangeControllers behavior, AdaptiveAgent uses it to get its controller, set data input, and then obtaining the calculated output.

The class diagram depicted in Figure 5 illustrates the controllers classes. Agents whereas virtualize homogeneous devices can use the same controller to make decisions. For example, in a scenario where similar smart lamps have to be managed, the same ANN controller can be used by Adaptive Agents. The GodAgent stores their controller in ControllerList as "lampNeuralNetwork". If there is another group of devices, the GodAgent has to attribute a different controller for them.

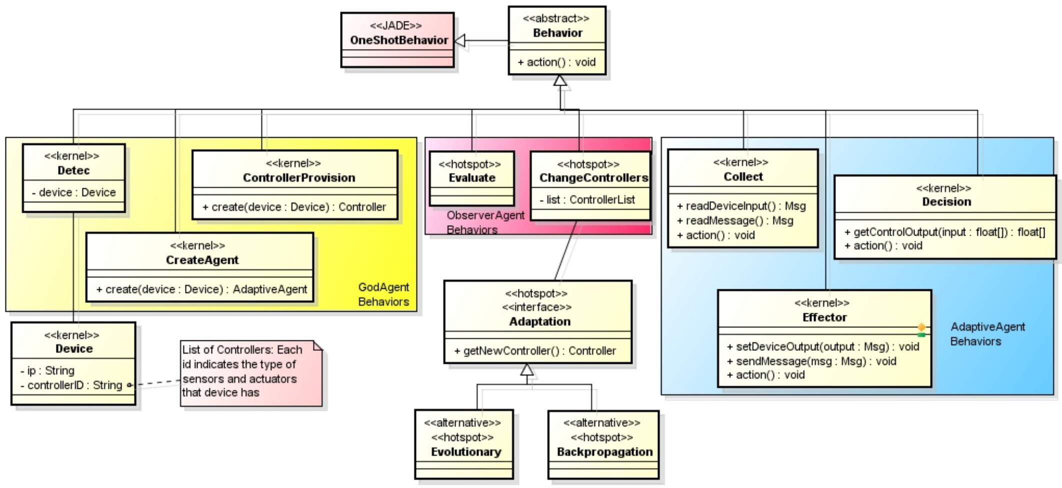
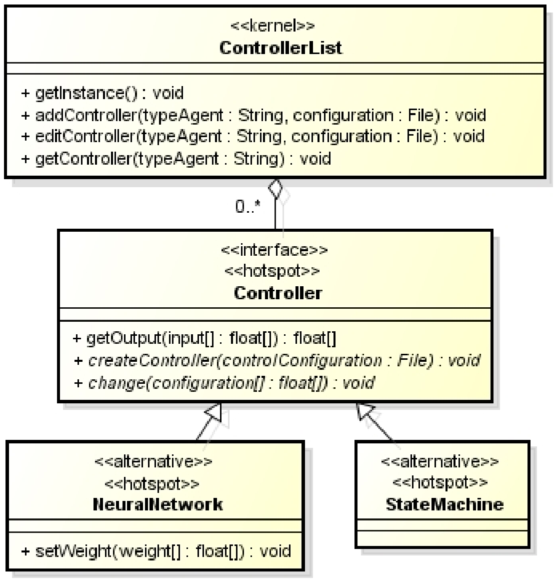


Figura 4: Class diagram of FIoT - Behaviors.



<<kernel>>
ControllerList
+ getInstance() : void
+ addController(typeAgent : String, configuration : File) : void
+ editController(typeAgent : String, configuration : File) : void
+ getController(typeAgent : String) : void
0..*
<<interface>>
<<hotspot>>
Controller
+ getOutput(input[] : float[]) : float[]
+ createController(controlConfiguration : File) : void
+ change(configuration[] : float[]) : void
<<alternative>>
<<hotspot>>
NeuralNetwork
+ setWeight(weight[] : float[]) : void
<<alternative>>
<<hotspot>>
StateMachine

Figura 5: Class diagram of FIoT - Controllers.

4 Evaluation: Illustrative Examples

We evaluate FIoT by implementing the flexible points of it to generate different applications. As discussed in the section 3, the framework instantiation is the last stage to be considered during the development of a framework [35].

We consider the following IoT instances in FIoT evaluation process: (i) Quantified Things and (ii) Smart City. For each one, we developed an illustrative example, and this section presents a brief description of them. We also show how the generated application adhere to the proposed framework, filling the main variable parts.

4.1 FIoT's Instances

The frozen spots are part of FIoT kernel. Then each of the proposed applications will have the following modules in common:

- Detection of devices by the GodAgent;
- The assignment of a controller to a particular Adaptive Agent by the GodAgent;
- Creation of Agents;
- Data Collection execution by Adaptive Agents;
- Making decision by Adaptive Agents;
- Execution of effective activity by Adaptive Agents;
- The communication structure among agents and devices.

Some features are variable and may be selected/developed according to the application type, as follows:

- Controller creation;
- Making evaluation by the ObserverAgent;
- Controller adaptation by the Observer Agent.

Therefore, to create a FIoT instance, a developer has to implement/choose: (i) a control module (e.g. neural network, finite state machine); (2) an adaptive technique to train the controller; and (iii) an evaluation process (e.g. genetic algorithm performs evaluation via fitness function). As shown in this section, we only evaluate applications using a neural network. However, we implemented FIoT to give support for the use of finite state machines (fsm), since we provided an abstract controller class. A framework user can implement a Mealy machine (a special case of a fsm), for example, and use an evolutionary algorithm to evolve its structure and transition probabilities [50, 34]. Thus, it is possible to generate applications using different configurations. A framework user needs to select a configuration that works better for solving a given problem.

4.2 Quantified Things

A new trend in Internet of Things is "Quantified Self"[51, 52], bringing the idea of continuous self-tracking. A person, equipped with sensors, allows his data to be available on the Internet and can monitor and evaluate his health information, for example. Since this information is already available, the community started to ask which kind of inferences is possible to make if selected groups of people share their tracked data, then the "Quantified Us"movement appeared [53, 52].

What happens if instead of asking "What can **people** learn when pooling data among themselves?"[54], we start to ask, "What can **things** 'learn' when pooling data among themselves?". ? These things could be factory machines, where one machine could predict a fault based on collective data shared among them. These things could also be bean plantations, where the owner of one plantation can predict the crop yield based on the history of crop from other bean plantations. Thus, a new branch of the quantify movement is proposed in this work, named "Quantified Things". In addition, a solution to design these kinds of experiments is presented using FIoT, as depicted in Figure 6.

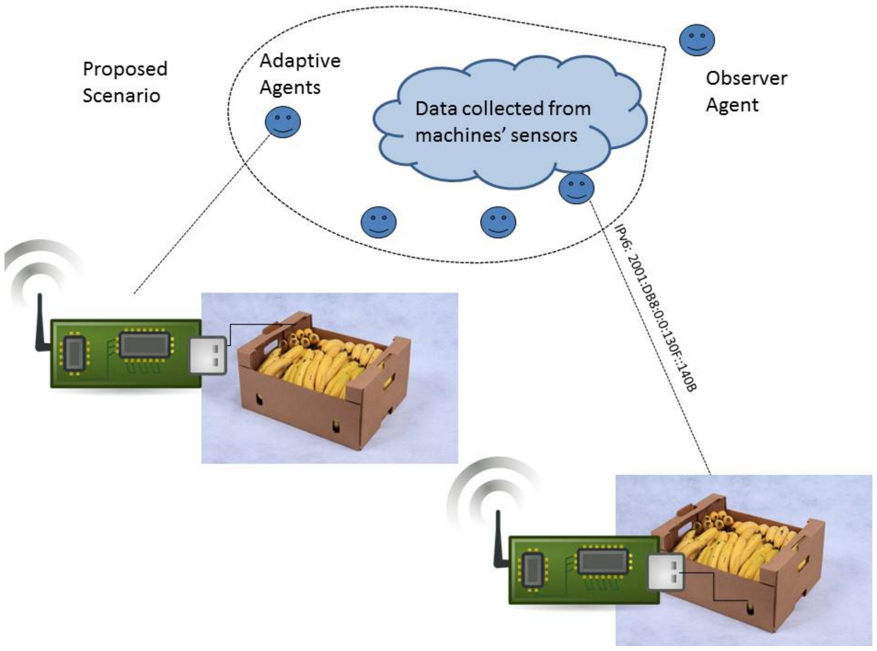

Proposed Scenario
Adaptive Agents
Data collected from machines' sensors
Observer Agent
IPv6: 2001:DB8:0:0:130F::1408

Figura 6: An instance of FIoT to create "Quantified Things"Instances.

Devices scattered across different environments are managed by adaptive agents. In turn, these agents populate a cloud database with sensored data from devices and their inferences. If new devices connect to the system, they can access this database and make predictions based on this historical data.

Quantified Bananas

The chosen scenario to make an instance of "Quantified Things"is "Quantified Bananas."In this example, stored data about similar bananas share data to predict how many days the bananas will spoil under specific environmental conditions. The data tracked from each bananas store are the temperature, humidity, luminosity and some gas sensors, as methane and hydrogen. The physical layer of this experiment scenario is depicted in Figure 7.

Adaptive agents use a three-layer feedforward neural network to predict the number of days to spoil. The sensored data from a device is used as the input of this three-layer network.

The ObserverAgent monitors executions, checking if prediction have been correct or not. To compare results, a user system needs to inform how long the monitored banana lasted. If results are not similar, the ObserverAgent executes the adaptation process to adjust the network parameters. The technique used is a supervised learning (backpropagation) since it has historical data to compare results and predict a new one.

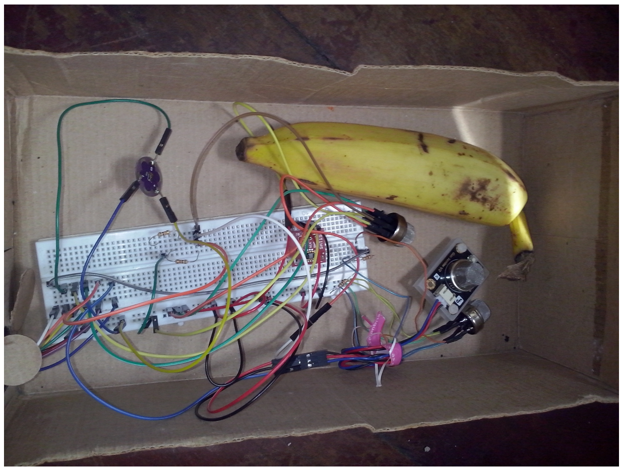


Figura 7: Physical layer for "Quantified Bananas"instance.

The Table 4.2 shows how the "Quantified Banana"application adhere to the proposed framework, extending the FIoT flexible points. The spot of "making evaluation"is developed for this applications as individual evaluation. The Observer Agent maintains a dataset containing input from Adaptive Agents and neural predictions. Based on this historical data, for each agent execution, the Observer Agent evaluates if individual result will generate a collective adaptation or not.

Tabela 2: Case I: Flexible Points

Framework	***Application***
Controller	Three Layer Neural Network
Making Evaluation	Individual Evaluation: for each agent evaluation, the Observer Agent concludes if all Adaptive Agents need to adapt or not
Controller Adaptation	Supervised Learning (Backpropagation)

4.2.1 Experimental Description

We carefully selected the individual bananas for haven a similar look. Each experiment was executed in a different condition. The experiments were created combining four parameters, as shown in Table 3: (i) dark (i.e. in a closed or open box); (ii) fridge (i.e. in the fridge or at room temperature); (iii) rotten fruit (i.e. put together with rotten fruit or not); and ripe fruit (i.e. put together with ripe fruit or not).

For example, in the first experiment, we placed a banana in an open box (not dark), at room temperature, and by itself. The ninth experiment was executed in a dark place, in the fridge, and together a rotten fruit.

Tabela 3: Experimental Description

Experiment	***Dark***	***Fridge***	***Rotten Fruit***	***Ripe Fruit***
1				
2				X
3			X	
4			X	X
5	X			
6	X			X
7	X		X	X
8	X		X	
9	X	X	X	
10		X	X	

4.2.2 Training Results

We verified the training process getting the predictions and comparing them with values registered as actual fruit shelf life for each experiment. This comparison is shown on Table 4, where the column “Expected Results”shows the “actual”shelf life, the column “Real Results”shows the predictions provided by the neural network, and the column “Error”the difference between these values, based on the normalized values.

Tabela 4: Results of backpropagation execution

Experiment	***Expected Results***	***Actual Results***	***Error***
1	1.0	0.999	$\approx$ 0
2	0.857	0.866	-0.0095
3	0.35	0.340	0.01
4	0.357	0.382	-0.025
5	0.714	0.719	-0.005
6	0.642	0.614	0.028
7	0.214	0.207	0.006
8	0.214	0.225	-0.010
9	0.285	0.285	$\approx$ 0
10	0.428	0.428	$\approx$ 0

As shown on Table 4, differences between expected and actual results are not so far off. The largest errors were presented in experiments four and six, corresponding to approximately one day. Both tests were executed at room temperature and with ripe fruit inside the box. A possible solution to reduce this error is to provide new experiments with similar settings, since the backpropagation algorithm needs an extensive dataset to train neural networks.

4.3 Smart City

Smart Cities is currently the hottest trend associated with the Internet of Things [55, 56, 3]. In order to have a Smart City it is fundamental to have many sensors scattered throughout the whole city collecting information, such as water and energy consumption, traffic and garbage monitoring.

In order to support smart services, IoT principles are applied to create self-managing car traffic control applications [57, 58], aiming at rebuilding the actual structure of traffic lights. For instance, cars, traffic lights and pedestrians will all be connected via the Internet, collecting and sharing data, such as GPS data from cars, traffic lights intervals, and camera images [58]. Based on this data, traffic lights will turn green or red, GPS consoles will offer drivers different routes, etc.

The reduction of urban traffic congestion continues to be the main goal of this new smart approach of car traffic management. For example, [59] propose that optimized traffic policies should be determined by the use of autonomous cars. However, given that their research focus on a intersection control mechanism, they only analyze how different policies affect a small portion of the road network.

According to Standford-Clark, an IBM engineer, the problem is not to change the traffic lights, but the "interconnection of unintended consequences."Thus, most traffic lights sequences are set via longer term algorithms, taking the whole of the road network into account [58]. Unfortunately, determining such sequences is a non-trivial and time consuming task, as one must take into account a wide range of factors like, traffic density, pedestrian flows, and road complexity.

FIoT makes it possible to create dynamic controllers for homogeneous things situated in a distributed environment by using a self-developing decentralized and adaptive process.

4.3.1 Car Traffic Application

In this subsection, we describe a simulated car traffic scenario, that stands as our application physical layer. Figure 8 depicts the elements that are presented in our scenario: vehicles, traffic lights, road segments, dividers and intersections. All roads are one-way; a segment is a portion of a road; intersections connect two or more segments; and a road divider divides a segment into two segments. We modeled our scenario as a graph, in which the edges represent segments and nodes represent road dividers and intersections.

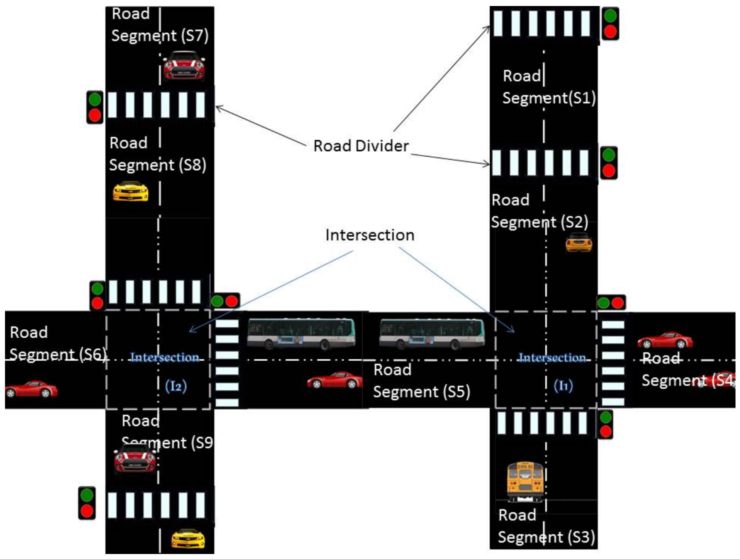


Figura 8: Traffic elements.

Smart Road Segment

Each road segment has a simulated microcontroller board associated with it that has an apparatus for calculating the rate of vehicles, interacting with the closest segment, and changing its own traffic light color.

Thus, the GodAgent creates an Adaptive Agent for each road segment in the scenario. Independently of the application, an Adaptive Agents always has to execute three tasks: data collection, decision making and action enforcement. For this experiment, the first task consists of receiving data collected by the respective road segment's microcontroller. It provides data related to vehicle flow, information from its neighbor segment and its current traffic light color.

To make decisions, Adaptive Agents use a "three-layer feedfoward"with a feedback loop (see [60]). Feedback occurs because the output of its traffic light color influences its next network input, as shown in Figure 9.

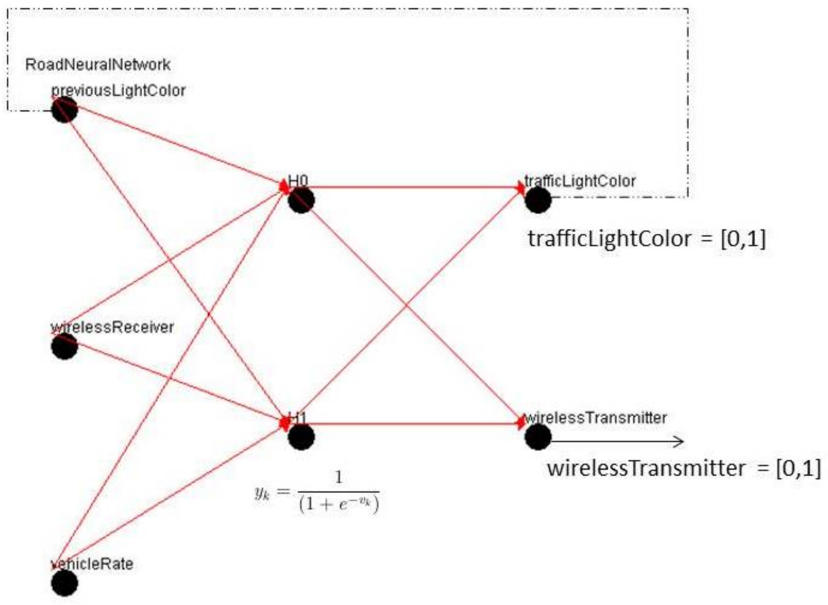


Figura 9: Adaptive Agent's Neural Controller.

By using a recurrent network, we aim at providing a kind of memory for these agents. Thus, our goal is to enable them to consider the duration span of a traffic light in a specific color. Therefore, the input layer consists of three neurons. The middle layer of the neural network has two neurons to connect the input and output layers. These neurons provide associations among sensors and actuators. These associations represent the system policies changing according to the encoded neural network configuration.

ObserverAgent: Adaptive process

Evolutionary algorithms have been applied to provide the design of system features automatically. By using a genetic algorithm, we expect that a light policy, sporting a simple communication system among road segments, will emerge from this experiment. Therefore, no system feature was specified at design-time (e.g. a communication system, the effect of vehicle rate on road segment decision). The evaluation and adaptation process performed by the Observer Agents is depicted in Figure 10.

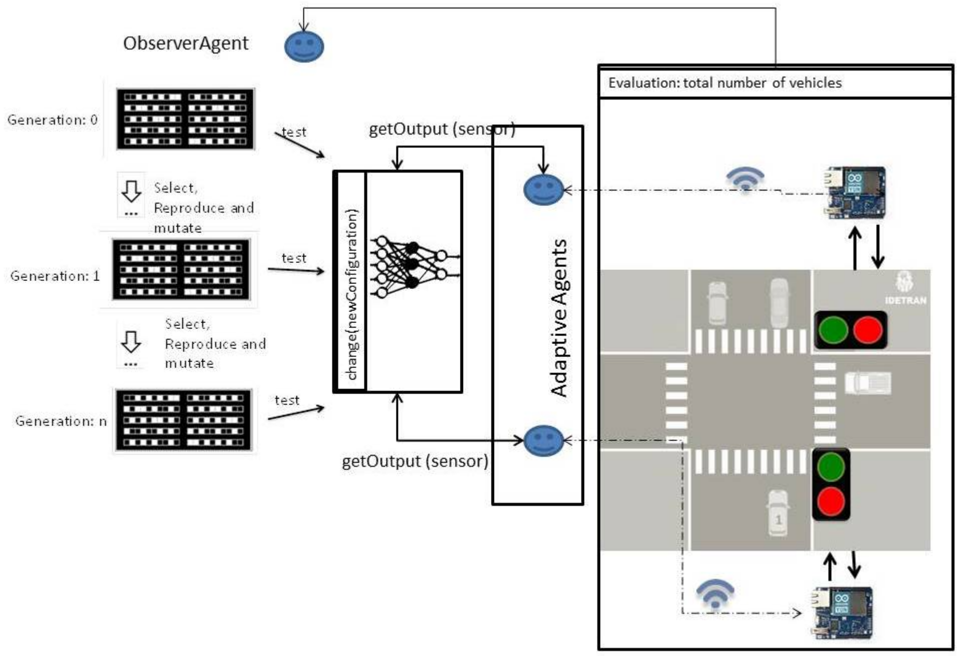


Figura 10: Performing an adaptive process to ajust the Traffic Neural Controller weights. Figure adapted from [61]. P.7.

The weights of the neural network used by the Adaptive Agents vary during the adaptive process. The ObserverAgent applies a genetic algorithm to find a better solution. It contains a pool of candidates to represent the network parameters. The ObserverAgent evaluates each one of them according to the number of cars that concluded their routes after the end of the simulation.

The Table 4.3.1 summarizes how the “Car Traffic Control”application adhere to the proposed framework, extending the FIoT flexible points.

Tabela 5: Case II: Flexible Points

Framework	***Application***
Controller	Three Layer Neural Network
Making Evaluation	Collective Fitness Evaluation: Test a pool of candidate to represent the network parameters. For each candidate, it evaluates the collection of Adaptive Agents, comparing fitness among candidates
Controller Adaptation	Evolutionary Algorithm: Generate a pool of candidate to represent the network parameters

Experiment

The first simulation scenario is depicted in Figure 11. We have created the urban road network scenario based on a small section of a real city, Feira de Santana, Bahia, Brazil. This chart is composed of 31 nodes and 48 segments. Each segment links two nodes having only one-way direction. For simulation purpose, we established 15 nodes as departures (yellow points) and two as targets (red points). Each segment has a traffic light. In the graph, the green and red triangles represent the traffic light colors.

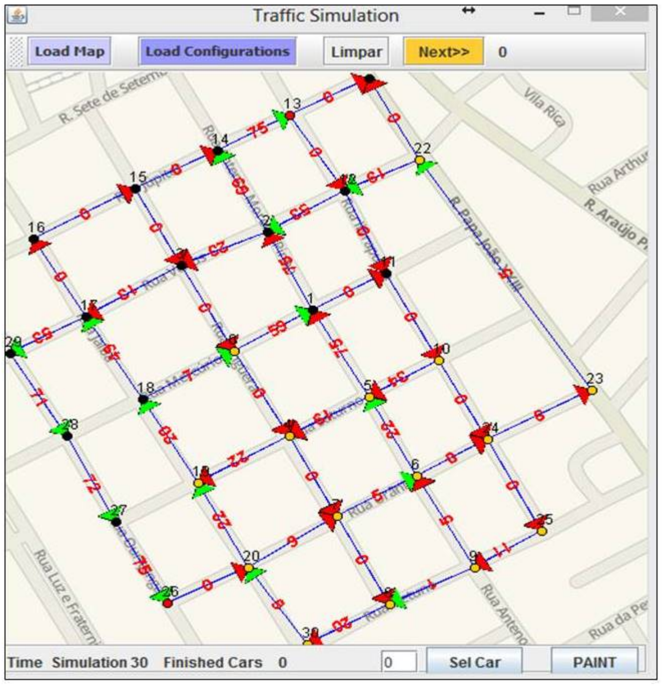


Figura 11: Simulation Urban Road Network. Adapted from Waze (2014) [62].

We started with 1000 vehicles for this experiment. The capacity of each road segment in this experiment is 75 vehicles. As we described before, the only role of the vehicles is to try to end their routes.

Evolutionary Algorithm: Simulation Parameters

Given that we are proposing a simple experiment, the evolutionary process lasts only 20 generations (i.e. the process of testing, selecting and reproducing candidates is iterated 20 times). During the test stage, each team of 48 Adaptive Agents (i.e. the number of road segments in the scenario) is allowed to "live"for 30 cycles by using a candidate, as shown in Figure 10. As each car departure and target are randomly selected and can affect the test result, more than one test is performed for each candidate.

The fitness of each candidate consists of the number of vehicles that concluded their route after the simulation ended. The individuals with the highest fitness are selected to generate the new generation by using crossover and mutation.

Evaluation of the Best Candidate

After executing the evolutionary algorithm, Adaptive Agents evolve the ability to find a satisfactory logical of traffic light decision in order to improve urban traffic flow.

We selected the best candidate from the evolutionary process to provide comparisons between our approach and "conventional"traffic light policies. We have been considering as conventional the normal way to control traffic lights, the so called fixed-time control. This type of control fixes the sequence of phases (red or green) and their durations [63]. We simulated two fixed-time approaches. The former changes all traffic lights colors in every cycle. The latter changes all the traffic lights colors at the intersections every two cycles, and sets the others green for 5 cycles and then red for only one cycle.

We executed the simulation three times, using the best solution presented above and each one of the two "conventional"solutions. Figure 12 presents the number of vehicles that concluded their routes.

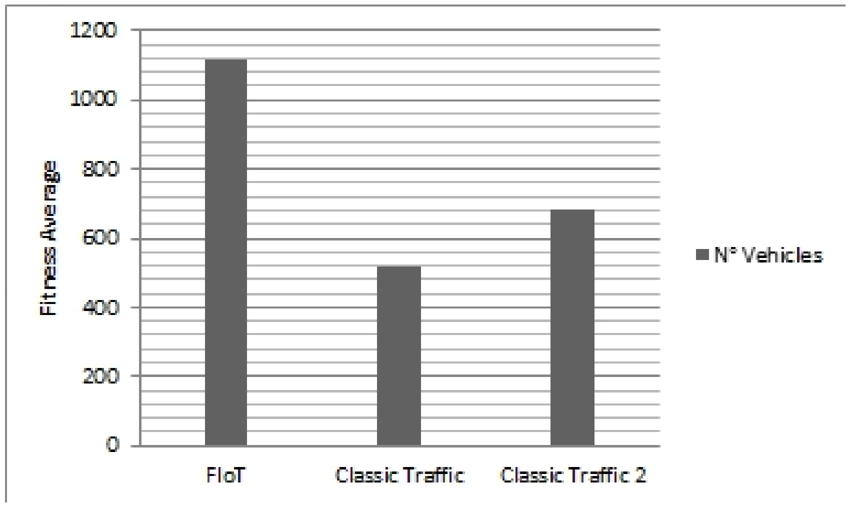


Figura 12: Comparison of the FIoT approach and conventional systems in the first scenario.

By using the evolved agents approach, the number of vehicles that concluded their routes is higher than using other approaches.

5 Conclusion

The area of Agent Oriented Software Engineering (AOSE) has so far addressed only small scale and even toy applications. We showed in this paper that a self-organizing and adaptive multi-agent software framework can be designed and implemented to derive now-a-days complex applications while doing so in an effective way. Software frameworks are domain oriented and we have chosen the Internet of Things (IoT) as such domain.

IoT applications are increasingly complex and require scalability beyond billions of devices and the ability to cope with environments that are in continuous transition. IoT is an emergent technology which has the potential for significant impact [64]. Self-organizing and self-adapting IoT multi-agent applications are an important evidence to show that agent oriented IoT assistants are an original contribution to both AOSE and IoT. The breath and relevance of MAS associated with Software Engineering to solve real world problems is the evidence we used to support our above claims.

We know of few research results in the literature about agent-based architectures for Internet of Things [14, 9]. None of them presents the design of a complex case study (i.e. using a vast number of cooperative things). Thus these works do not show efficiency in wide scenarios, where things must cope with a changing environment and where a sophisticated organization system is required. Therefore, important features mentioned on our work regarding self-organization and self-adaptation are not covered by the related literature. On the other hand, we found several effective experiments in the Robotic Agents literature about complex autonomous physical systems (as swarm of robots). We used in our approach assumptions made by those studies regarding self-adaptive and self-organizing properties for physical agents domain.

We provided two instances of our proposed agent-based framework for the IoT domain: (i)quantified bananas; and (ii)traffic light control. Through those instances we have showed that our agent-based general software system satisfies its main goals:

- Autonomous things;
  - Things that are able to cooperate and execute complex behavior without the need for centralized control to manage their interaction.
  - Things that are able to have behavior assigned at design-time and/or at run-time.
- Feasible modelling characteristics;
  - It is possible to use our framework model to deal with complex problems in considerable time.
  - In particular, it is possible during the design phase, that one does not need to be concerned with the application domain.

6 Future Works

We believe that as FIoT matures, it will be able to support the development of more complex and realistic IoT applications, especially in actual distributed environments. As future work, we want to investigate the generalization capacity of our proposed framework and its application limits. As such, we need to evaluate FIoT by taking the following criteria into account:

- To investigate different applications types from prediction and control to discover which IoT application types can be created by using FIoT;
- To investigate different techniques from back-propagation and genetic algorithm to discover which types of adaptive techniques can be used;
- To evaluate which types of control can be used to meet the requirements imposed by the FIoT's controller abstract class. In addition, we have to investigate the adaptive techniques that are suitable with each proposed control.

- To investigate which IoT applications require adaptation and the types of (self)adaptations that are useful for them.

As a result, new FIoT requirements and hot spots can appear. For example, a further application may require the management of heterogeneous environments and devices. Thus, to enable the production of new instances, we will probably need to increase the FIoT domain coverage and create new hot spots.

The centralized architecture is the major problem on God Agent specifications. There is only one God for each application. It may be a problem for applications that requires multiples devices connecting simultaneously. In future works, we can investigate existing discovery services architectures to provide FIoT applications with self-discovery protocols and more scalability.

Acknowledgements

This work has been supported by the Laboratory of Software Engineering (LES) at PUC-Rio. Our thanks to CNPq, CAPES, FAPERJ and PUC-Rio for their financial support.

References

[1] P. Rodrigues, Y.-D. Bromberg, L. Reveillere, D. Negru, Zigzag: A middleware for service discovery in future internet, in: Distributed Applications and Interoperable Systems, Springer Berlin Heidelberg, 2012, pp. 208–221.

[2] J. Park, L. Barolli, F. Xhafa, H. Jeong, Information Technology Convergence: Security, Robotics, Automations and Communication, Lecture Notes in Electrical Engineering, Springer, 2013.
URL `https://books.google.com.br/books?id=6sbEBAAAQBAJ`

[3] J. Gubbia, R. Buyyab, S. Marusic, M. Palaniswami, Internet of things (iot): A vision, architectural elements, and future directions, Future Generation Computer Systems 29 (2013) 1645–1660.

[4] L. Atzori, A. Iera, G. Morabito, The internet of things: A survey, Computer networks 54 (15) (2010) 2787–2805.

[5] G. Fortino, P. Trunfio, Internet of Things Based on Smart Objects: Technology, Middleware and Applications, Springer, 2014.

[6] F. Rochner, H. Prothmann, J. Branke, C. Müller-Schloer, H. Schmeck, An organic architecture for traffic light controllers., in: GI Jahrestagung (1), 2006, pp. 120–127.

[7] E. Velloso, A. Raposo, H. Fuks, Web of things: The collaborative interaction designer point of view, in: 1st Workshop of the Brazilian Institute for Web Science Research, 2010.

[8] D. Bandyopadhyay, J. Sen, Internet of things: Applications and challenges in technology and standardization, Wireless Personal Communications 58 (1) (2011) 49–69.

[9] P. Lopez, G. Perez, Collaborative agents framework for the internet of things, in: Ambient Intelligence and Smart Environments, 2012, pp. 191–199.

[10] S. Beydeda, M. Book, V. Gruhn, Model-Driven Software Development, Springer-Verlag Berlin Heidelberg, 2005.

[11] I. Sommerville, Software Engineering, International computer science series, Pearson/Addison-Wesley, 2004.
URL `http://books.google.com.br/books?id=fIJQAAAAMAAJ`

[12] G. Fortino, A. Guerrieri, W. Russo, Agent-oriented smart objects development, in: IEEE International Conference on Computer Supported Cooperative Work in Design (CSCWD), 2012.

[13] G. Fortino, A. Guerrieri, W. Russo, C. Savaglio, 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. 1–29.

[14] G. Fortino, A. Guerrieri, M. Lacopo, M. Lucia, W. Russo, An agent-based middleware for cooperating smart objects, in: Highlights on Practical Applications of Agents and Multi-Agent Systems, Springer Berlin Heidelberg, 2013, pp. 387–398.

[15] G. Di Marzo Serugendo, M.-P. Gleizes, A. Karageorgos, Self-organization in multi-agent systems, The Knowledge Engineering Review 20 (02) (2005) 165–189.

[16] D. Marocco, S. Nolfi, Emergence of communication in embodied agents evolved for the ability to solve a collective navigation problem, Connection Science.

[17] D. Floreano, C. Mattiussi, Bio-Inspired Artificial Intelligence. Theories, Methods, and Technologies, Cambridge: MIT Press, 2008.

[18] G. Weiss, S. Sen, Adaptation and Learning in Multi-Agent Systems,, Springer-Verlag, 1995.

[19] G. Di Marzo, A. Karageorgos, O. Rana, F. Zambonelli, Engineering Self-Organising Systems, Springer, Berlin, 2004.

[20] K. Cetnarowicz, K. Kisiel-Dorohinicki, E. Nawarecki, The application of evolution process in multi-agent world to the prediction system, in: Second International Conference on Multiagent Systems, 1996, pp. 26–32.

[21] M. Quinn, L. Smith, G. Mayley, P. Husbands, P. H. Nds, Evolving controllers for a homogeneous system of physical robots: Structured cooperation with minimal sensors (2003).

[22] L. Steels, Ecagents: Embodied and communicating agents, Tech. rep., SONY (2004).

[23] V. Trianni, S. Nolfi, Engineering the evolution of self-organizing behaviors in swarm robotics: A case study, Artificial Life 17 (3) (2011) 183–202.

[24] S. Nolfi, D. Floreano, Evolutionary Robotics: The Biology,Intelligence,and Technology of Self-Organizing Machines, MIT Press, Cambridge, MA, USA, 2000.

[25] A. Nelson, G. Barlow, L. Doitsidis, Fitness functions in evolutionary robotics: A survey and alasysis, Robotics and Autonomous Systems.

[26] F. Kawsar, T. Nakajima, J. Hyuk Park, S. Yeo, Design and implementation of a framework for building distributed smart object systems, Supercomputing.

[27] C. Goumopoulos, A. Kameas, Smart objects as components of ubicomp applications, International Journal of Multimedia and Ubiquitous Engineering.

[28] M. Muhlhauser, Smart products: An introduction, Communications in Computer and Information Science.

[29] G. Pezzulo, G. Baldassarre, A. Cesta, S. Nolfi, Research on cognitive robotics at the institute of cognitive sciences and technologies, national research council of italy, Cognitive processing 12 (4) (2011) 367–374.

[30] G. Massera, T. Ferrauto, O. Gigliotta, S. Nolfi, Farsa: An open software tool for embodied cognitive science, in: Advances in Artificial Life, ECAL, Vol. 12, 2013, pp. 538–545.

[31] S. Nolfi, D. Parisi, Learning to adapt to changing environments in evolving neural networks, in: Adaptive Behavior, 1997, pp. 75–98.

[32] S. Nolfi, D. Floreano, Co-evolving predator and prey robots: Do 'arms races' arise in artificial evolution? (1998).

[33] G. Massera, T. Ferrauto, O. Gigliotta, S. Nolfi, Designing adaptive humanoid robots through the farsa open-source framework, Tech. rep., Institute of Cognitive Sciences and Technologies (CNR-ISTC) (2013).

[34] A. Sobe, I. Fehervari, W. Elmenreich, Frevo: A tool for evolving and evaluating self-organizing systems, in: IEEE Self-adaptive and Self-organizing Systems Workshop, 2012.

[35] M. E. Markiewicz, C. J. P. de Lucena, Object oriented framework development, Crossroads 7 (4) (2001) 3–9. doi:10.1145/372765.372771.
URL http://doi.acm.org/10.1145/372765.372771

[36] T. von der Maßen, H. Lichter, Modeling variability by uml use case diagrams, in: Proceedings of the International Workshop on Requirements Engineering for product lines, Citeseer, 2002, pp. 19–25.

[37] M. Dumas, A. ter Hofstede, Uml activity diagrams as a workflow specification language, in: UML 2001 â The Unified Modeling Language. Modeling Languages, Concepts, and Tools, Springer Berlin Heidelberg, 2001, pp. 76–90.

[38] G. D. M. Serugendo, J. Fitzgerald, A. Romanovsky, N. Guelfi, A generic framework for the engineering of self-adaptive and self-organising systems, University of Newcastle upon Tyne, Computing Science, 2007.

[39] M. Kuniavsky, Smart Things: Ubiquitous Computing User Experience Design Book, Morgan Kaufmann, 2010.

[40] F. Bellifemine, G. Caire, T. Trucco, G. Rimassa, R. Mungenast, Jade Administrator's Guide, JADE, jade.tilab.com/doc/administratorsguide.pdf (2007).

[41] F. Bellifemine, G. Caire, T. Trucco, G. Rimassa, Jade Programmerâs Guide, jade.tilab.com/doc/programmersguide.pdf (April 2010).

[42] A. J. Riel, Object-oriented design heuristics, Vol. 335, Addison-Wesley Reading, 1996.

[43] C. Müller-Schloer, Organic computing: on the feasibility of controlled emergence, in: Proceedings of the 2nd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis, ACM, 2004, pp. 2–5.

[44] M. Wooldridge, An introduction to multiagent systems, John Wiley & Sons, 2009.

[45] B. Neto, A. Costa, M. Netto, V. Silva, C. Lucena, Jaaf: A framework to implement self-adaptive agents, in: International Conference on Software Engineering and Knowledge Engineering, 2009.

[46] P. Horn, Autonomic computing: Ibm\'s perspective on the state of information technology.

[47] B. Jacob, R. Lanyon-Hogg, D. K. Nadgir, A. F. Yassin, A practical guide to the ibm autonomic computing toolkit (2004).

[48] M. Dorigo, V. Trianni, E. Şahin, R. Groß, T. H. Labella, G. Baldassarre, S. Nolfi, J.-L. Deneubourg, F. Mondada, D. Floreano, et al., Evolving self-organizing behaviors for a swarm-bot, Autonomous Robots 17 (2-3) (2004) 223–245.

[49] L. Panait, S. Luke, Cooperative multi-agent learning: The state of the art, Autonomous Agents and Multi-Agent Systems 11 (3) (2005) 387–434. doi:10.1007/s10458-005-2631-2.
URL `http://dx.doi.org/10.1007/s10458-005-2631-2`

[50] A. Pintér-Bartha, A. Sobe, W. Elmenreich, Towards the lightâcomparing evolved neural network controllers and finite state machine controllers, in: Intelligent Solutions in Embedded Systems (WISES), 2012 Proceedings of the Tenth Workshop on, IEEE, 2012, pp. 83–87.

[51] D. Rose, Enchanted Objects: Design, Human Desire, and the Internet of Things, Scribner, 2014.
URL `https://books.google.com.br/books?id=8QIGAgAAQBAJ`

[52] M. Jordan, N. PFARR, Forget the quantified self. we need to build the quantified us, http://www.wired.com/2014/04/forget-the-quantified-self-we-need-to-build-the-quantified-us/ (April 2014).

[53] V. Lee, Learning Technologies and the Body: Integration and Implementation In Formal and Informal Learning Environments, Routledge Research in Education, Taylor & Francis, 2014.
URL `https://books.google.com.br/books?id=iVacBQAAQBAJ`

[54] J. Havens, Hacking Happiness: Why Your Personal Data Counts and How Tracking it Can Change the World, Penguin Publishing Group, 2014.
URL `https://books.google.com.br/books?id=rRQLZTnkUpYC`

[55] J. Bohli, P. Langendorfer, A. F. Skarmeta, Security and privacy challenge in data aggregation for the iot in smart cities, Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems (2013) 225–244.

[56] S. Mitchell, N. Villa, M. Stewart-Weeks, A. Lange, The internet of everything for cities: Connecting people, process, data, and things to improve the âlivabilityâ of cities and communities (2013).

[57] D. P. Möller, Introduction to Transportation Analysis, Modeling and Simulation, Springer, 2014.

[58] TheGuardian, Can the internet of things save us from traffic jams?, http://www.theguardian.com/technology/2015/apr/20/internet-of-things-traffic (April 2015).

[59] D. Carlino, M. Depinet, P. Khandelwal, P. Stone, Approximately orchestrated routing and transportation analyzer: Large-scale traffic simulation for autonomous vehicles, in: Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on, IEEE, 2012, pp. 334–339.

[60] S. Haykin, Neural Networks: A Comprehensive Foundation, Macmillan, 1994.
URL `http://books.google.com.br/books?id=PSAPAQAAMAAJ`

[61] S. Nolfi, O. Gigliotta, Evorobot*, in: Evolution of communication and language in embodied agents, Springer, 2010, pp. 297–301.

[62] W. MOBILE, Waze. disponível em:¡ https://www. waze. com/pt-br¿, Acesso em 2.

[63] H. Prothmann, S. Tomforde, J. Branke, J. Hähner, C. Müller-Schloer, H. Schmeck, Organic traffic control, Springer, 2011.

[64] C. Stamford, 2014 hype cycle for emerging technologies maps the journey to digital business, Tech. rep., Gartner, http://www.gartner.com/newsroom/id/2819918 (August 2014).