A Multi-agent Based Governance of Machine-to-Machine Systems

Camille Persson[1,2], Gauthier Picard[2], Fano Ramparany[1], and Olivier Boissier[2]

[1] Orange Labs Network and Carrier, TECH/MATIS, Grenoble, France
`firstname.lastname@orange.com`
[2] Fayol Institute, Ecole des Mines de Saint-Etienne, France
`lastname@emse.fr`

Abstract. In *Machine-to-Machine (M2M)* systems, multiple devices (sensors, actuators), situated in the physical world, interact together to provide data to added value services. In the *SensCity* project, the proposed M2M infrastructure, to support cityscale application, must be able to support the increase in the number of services and applications that are deployed on it. It is thus necessary to share the infrastructure use dynamically between them. In this paper, we propose the use of multi-agent technologies to define an adaptive and agile layer to govern and adapt the M2M infrastructure to the different applications using it. We illustrate our proposal within a smart parking management application.

1 Introduction

The next generation of cities are getting smarter by providing automated services to improve the life of their citizens (e.g. optimized garbage collection, smart metering, traffic redirection and parking management). These added value services build what we call *Machine-to-Machine (M2M)* systems, i.e. a network of smart devices –sensors and actuators– interacting with each other without human intervention (see Fig. 1).

Recent improvements in low-power wireless technologies [15] enable wireless devices to be connected to the Internet with a low deployment cost and a long lifespan –20 years expected. Such developments allow applications to be immersed in the real world and to directly act on the environment. When looking at the deployment of such

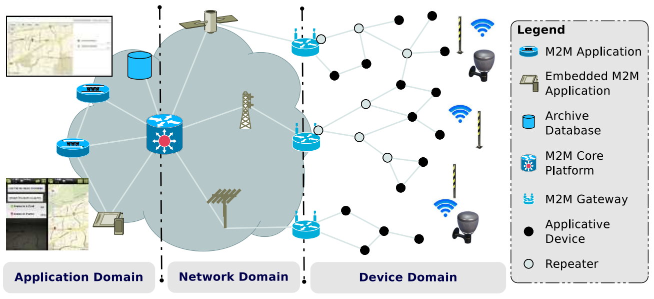

Fig. 1. An end-to-end M2M architecture [6]

systems, stakeholders are involved in different areas: application providers, constrained devices constructors, LLNs radio experts and telecommunication operators. However, the building of such *vertical* solutions on a city scale is too expensive and not flexible enough. There is a growing need and interest for M2M infrastructures that provide *horizontal* integration and sharing of devices between stakeholders [1].

This paper considers the practical use case issued from the SensCity[1] project, which aims at providing an M2M platform for deploying multiple smart city services. This platform makes possible to connect heterogeneous wireless devices to a GPRS gateway using Wavenis –long range, energy efficient radio technology. As for the application, standard access to the devices is provided via web services.

In order to be deployed on a city scale and used by different applications, such an infrastructure is faced with a multi-faceted problem of scalability [7] in terms of size, heterogeneity, topology, etc.To tackle this problem, an agile governance is required to conciliate both "vertical" and "horizontal" concerns. Such a governance system should adapt to the different requirements of the M2M applications deployed on such an infrastructure so that the whole system scales.

Given the complexity and inherent decentralization of the infrastructure, we propose the use of multi-agent technologies to define this governance layer on top of the M2M infrastructure. We used a newborn multi-agent oriented programming framework called JaCaMo[2] to implement it. In order to obtain an agile governance, the governance strategy is defined as an explicit organization specification, using the $\mathcal{M}OISE$ framework [9], part of the JaCaMo platform. Thanks to the explicit and agent-readable specifications, agents are able to reason about the governance of the system to change and adapt it to the evolution of the system. The proposed multi-agent governance is illustrated by a smart parking management application.

Section 2 motivates our approach with a description of the M2M infrastructure and the smart parking application that we consider in this paper. Based on this applicative context, we describe the multi-agent governance layer deployed on top of the M2M components (Section 3). We then focus, in Section 4, on the definition of the governance strategy stressing how it can be dynamically adapted by agents. The application of this governance applied to the smart parking management use case is described in Section 5. Then, Section 6 discusses the proposed approach and compares it to related works. Finally, Section 7 concludes this paper and considers the perspectives for future work.

2 Motivations

Machine to Machine (M2M) systems are an early technology which is just raising out of fully proprietary solutions with different standard proposals [6]. We first give an overview of the *M2M architecture* standard proposal on which we are basing our work. We then introduce the smart parking management application as an illustrative example where we highlight the need of an agile governance layer.

[1] The SensCity project (FUI Minalogic) Sensors and Services for Sustainable Cities: `http://www.senscity-grenoble.com/`

[2] `http://www.jacamo.sourceforge.net`

2.1 Machine-to-Machine Architecture

The M2M Technical Committee of the European Telecommunications Standards Institute (ETSI) is defining standards for M2M infrastructure. The scope of these standards cover communication from the devices to the applications, through gateways and a core platform. As shown in the latest version of ETSI's specification draft [6], the M2M architecture is divided into three domains (cf. Fig. 1): *Device*, *Network* and *Application*.

The Device Domain is composed of applicative devices –*sensors* and *actuators*– and repeaters communicating in a Wireless Sensor and Actuator Network (WSAN) linked to a gateway. The WSAN groups several devices communicating together. Devices can embed several sensors and actuators, or none of them. *Repeaters* are placed to extend the coverage managed by a *gateway*. It manages one or several WSANs, security and device authentication and can also manage quick reaction to sensed events generating commands to devices. The *gateway* sends/receives messages to/from the *platform* via broadband access. Thus it also belongs to the *Network Domain*. The *core platform* is involved in both the Network Domain and the Application Domain, as shown with the synthesis of the functionalities of this platform in Table 1. On one hand, it is responsible for network communications with other platforms. On the other hand, it gives the *application* –managing business logic– transparent access to the devices and stores the messages (see *ETSI M2M Functional Specification* [6] for further details).

Table 1. M2M core platform's functionalities [6]

Network Domain		Application Domain	
REM	Remote Entity Management	AE	Application Enablement
GC	Generic Communication	CB	Compensation Brokerage
RAR	Reachability Addressing and Repository	TM	Transaction Management
CS	Communication Selection	HDR	History Data Retention
IP	Interworking Proxy		
SEC	Security		

2.2 Use Case Scenario: A Smart Parking Management

In order to be deployed at a city scale and used by different client applications, a *vertical solution* as a whole cannot be used since it is too expensive and not flexible enough. Installing *horizontal integration* solution is of growing interest in order to share devices between different stakeholders [1]. To this aim, agility is required for conciliating different requirements.

In the following, let's consider a Smart Parking System where car detectors are used for monitoring parking places. The collected data are used and shared between, at least, two applications: Car Guidance, City Monitoring. When a car parks in or leaves the place, the event is notified through the M2M infrastructure: a message is 1. sent to the gateway (Device Domain) which 2. authenticates and 3. forwards it to the M2M platform (Network Domain) where 4a. it is stored and 4b. notified to subscriber applications (Application Domain), which in turn 5. retrieve the message. Applications can also send commands to the devices using the reverse path to act on the environment, e.g. to raise a parking post to reserve a place.

The *Car Guidance* application helps drivers to find a parking place directly and close to their destination following their preferences, reducing traffic flow and pollution[3]. To do this, it needs to monitor the places within an area around the destination when the driver is getting close to this area. In the case of reservable parking places, the application can send a message to actuate a parking post for the user.

The *City Monitoring* application is used by city services (eg. police) to monitor no-parking places. It requires alerts to be sent when a place is occupied during a nonstationary time with a variable priority (e.g. water access for firemen has priority over garage doors).

As sending messages consumes a lot of energy, due to several applications sharing the same devices with heterogeneous requirements, it raises issues such as scalability and energy consumption. Furthermore, other applications, using other devices, will share the same infrastructure –the platform and the gateways– generating traffic and resource management issues.

In this context, an agile governance is required to define how the resources –devices, servers, network– should be used. Each *vertical* requirements can be specified by a *Service Level Agreement (SLA)* contract between an *application*, the *shared infrastructure* (i.e. servers, gateways and repeaters) and a set of *devices*. The governance layer concerns both the *vertical* infrastructure –i.e. interactions between an application and devices– and the *horizontal* infrastructure –i.e. sharing the resources between different applications. The goal of this paper is to propose a multi-agent based governance model to manage the shared M2M infrastructure.

3 Overview of the Multi-agent Governance

Given the different requirements and motivations expressed in the previous section, this section describes the multi-agent approach used to define the M2M governance, describe by Fig. 2. To clearly separate the different concerns that arise in such applications, we have chosen a multi-agent oriented programming approach which is supported by the JaCaMo[4] [18] framework. This multi-agent oriented programming framework allows the development of MAS taking into account three different programming dimensions, namely *agent*, *environment*, and *organization*.

JaCaMo is built upon the synergistic integration of three existing agent-based technologies: *(i)* *Jason* [4], *(ii)* MOISE [9], and *(iii)* CArtAgO[19]. A JaCaMo multi-agent system (i.e., a software system programmed in JaCaMo) is given by a multi-agent organization programmed in MOISE, organizing autonomous agents programmed in *Jason*, working in shared distributed artifact-based environments programmed in CArtAgO. JaCaMo integrates these three platforms by defining a semantic link among concepts of the different programming dimensions at the meta-model and programming levels, in order to obtain a uniform and consistent programming model aimed at simplifying the combination of those dimensions when programming multi-agent systems.

[3] Parking search is estimated to be from 5% to 10% of traffic and represented a total waste of 70 millions hours for a cost of 600 millions in France [13] (2005).

[4] `http://www.jacamo.sourceforge.net`

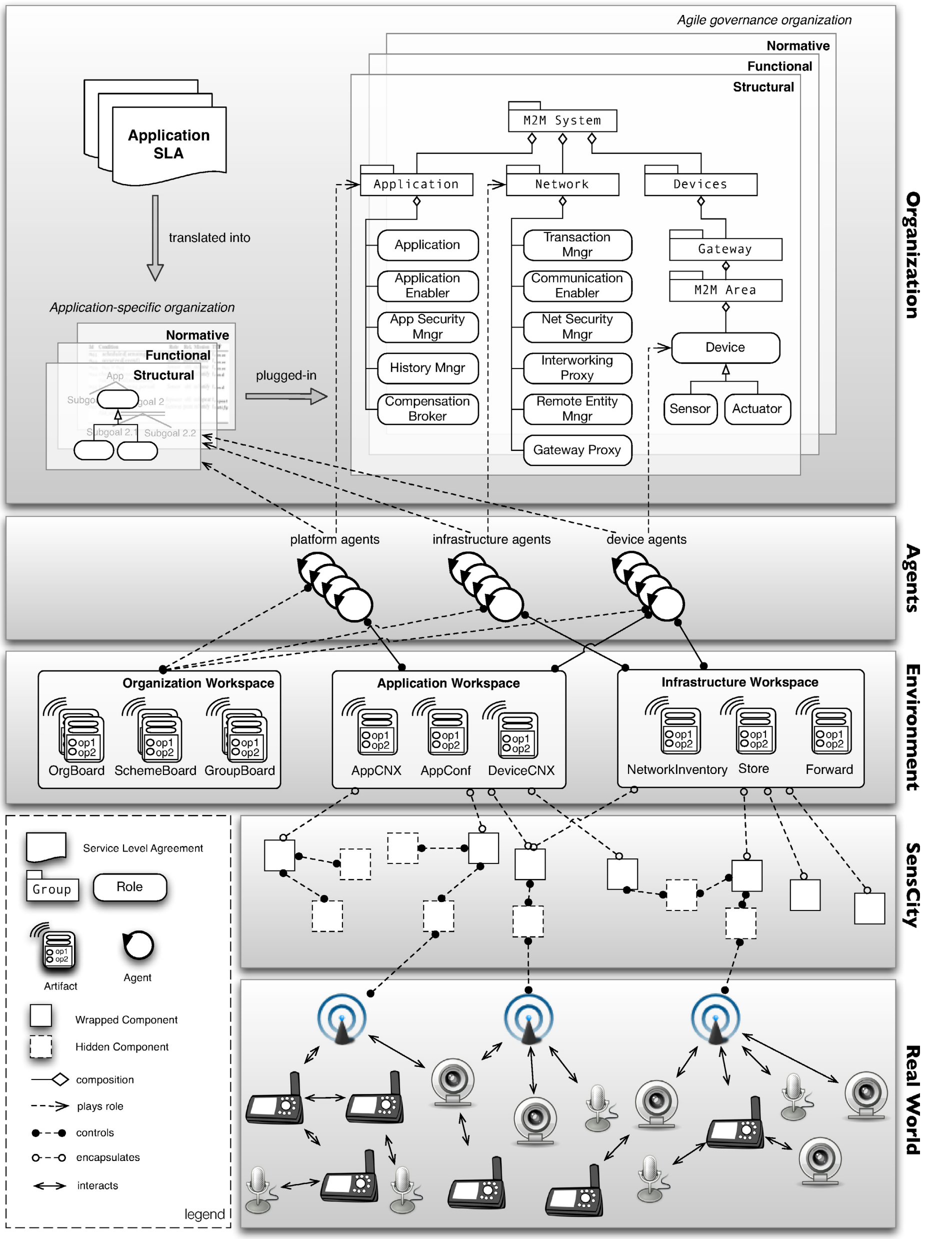

Fig. 2. On top of the *SensCity* *Core Platform*, the multi-agent based governance layer architecture is as follows: *artifacts* encapsulate the components to allow the *agents* to control the platform by applying the SLAs strategy defined by an *organization* following the ETSI recommendations

These three dimensions are used to define the governance layer deployed on top of the M2M infrastructure (cf. Fig. 2) aiming at governing its use by the different applications: (*i*) artifacts encapsulate the infrastructure components and provide the agents with the necessary actions and perceptions to monitor and control the use of these components, (*ii*) agents are the reasoning entities that make local decisions with respect to the governance taking into account their partial view on the infrastructure status and that cooperate with the other agents participating in the governance, (*iii*) organization that structures and regulates the autonomous functioning of the agents with respect to the global governance strategy defined from the requirements issued from the applications providing added value services in the smart city by acting on and consuming the data provided by the M2M infrastructure.

The organization limits the place of the possible actions that the agents can execute, letting them decide locally and autonomously. Thanks to the $\mathcal{M}$OISE framework, agents are able to reason about the organization and decide to change it when it is not adapted anymore to the current state of the governance requirements (e.g. high number of violations greater than authorized by the SLA contract).

Before detailling the governance strategy as an organization in the next section (Section 4), we first start by describing the use of artifacts (Section 3.1) to monitor and control the SensCity platform and the agents (Section 3.2) of the proposed governance model.

3.1 Artifacts to Control the M2M Infrastructure

Artifacts defined with the CArtAgO platform are used to encapsulate components of the M2M infrastructure to give the agents the control of it. In the context of the *SensCity* project, the governance layer is deployed on top the core platform which is divided into an *USP* part to manage the notifications sent to the applications, their rights and billing, and an *UCCP* part to manage the devices and communications with them. Fig. 3 describes the component-based architecture of these two platforms. Artifacts encapsulate the components' functionality.

These artifacts are used to give the agents a representation of the system to govern. An artifact monitors one or several components' activity: the agents are notified of statuses and calls to the components by signals and observable properties. The artifact's operations enable the agents to use the component.

3.2 Agents to Apply and Reason about the Governance Strategy

Agents are the decision-making entities of the governance layer. They adopt one or several roles in the organization corresponding to the part of the governance for which they assume responsibility. Following the strategy specification given by the organization (described in Section 4), they ensure that the M2M system is functioning correctly by monitoring the infrastructure and adapting it using the artifacts. For example, if an agent notices an overload on the platform, it can interrupt the calls to its components in order to filter the calls and redirect some of them to another platform.

Agents implements the governance policies. They can reason about the strategy definition and evaluate it with respect to the M2M system's functioning. They can adapt either the M2M infrastructure using the artifacts or redefine the strategy by proposing new organizational specifications.

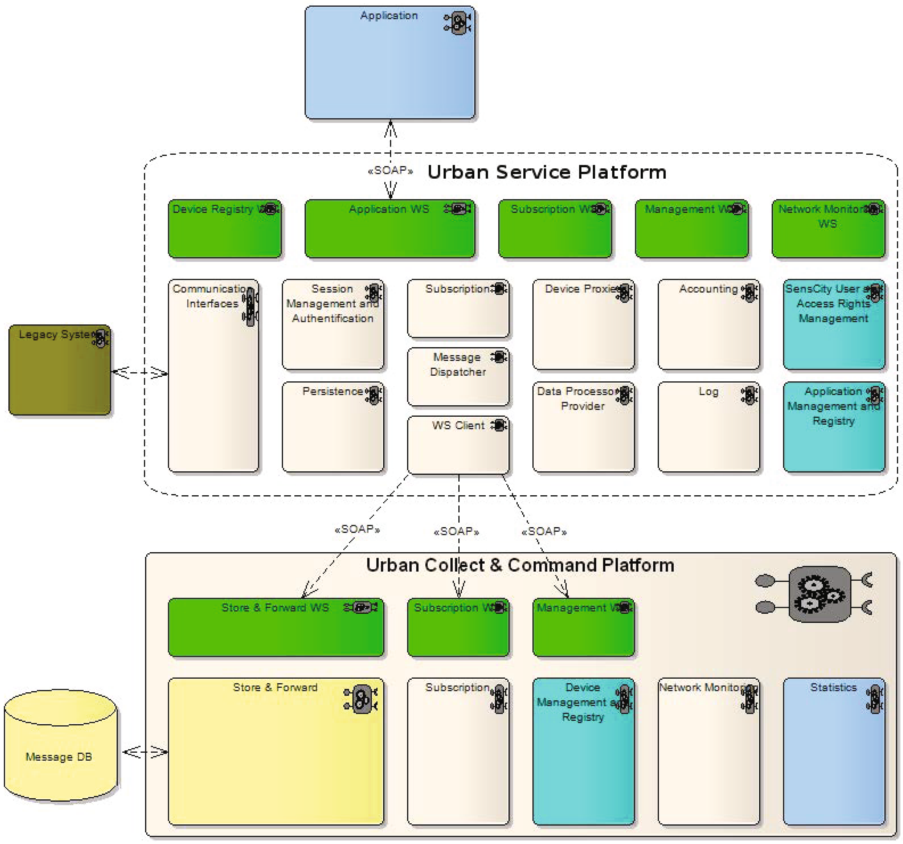

Fig. 3. The *SensCity's core platform* component-based architecture divided into an *USP* part communicating with applications and an *UCCP* part communicating with the gateways and devices

As the different parts of the M2M infrastructure should be handled differently, different kinds of agents have been identified: *appAg*, *platformAg*, *gwAg* and *deviceAg*.

The *appAg* agents ensure the requirements fulfilment from the application point of view. Thus, this type of agent regulates the commands and requests sent by an application to the devices and check the notifications it receives. To perform this regulation, an AppAg is able to intercept messages by using the *AppCNXArt* artifact to control them and validate the transmission.

Similarly, the *deviceAg* agents control the usage of the devices specified by the SLA, by using encapsulation artifacts such as *DeviceCNXArt*, to ensure that the devices perform their obligations. The agent can evaluate the load of a device by the number of roles it has adopted, and so make smarter use of the device (e.g. combine two messages at once, or skip messages if not necessary). The main goal of such an agent is to make the device's life as long as possible. It negotiates the requirements in this aim and can eventually give priority to one contract over another.

The *platformAg* agents are responsible for the platform functioning. They contribute to the contract agreement by evaluating the traffic and the load it will generate on the server itself. For example, when it is too high, it can intercept calls to some components and redirect them to a delegated server. This has to be done with respect to the latency requirements specified, so redirection has to be carried out according to the priority of the message and its destination. For example, priority to notifications to the *CityMonitoring* application over the *CarGuidance*.

The *gatewayAg* agents are concerned with traffic and load on the *Gateway*, as *platformAg*, but also with the local rule treatment. Indeed, *application* can define rules to generate –i.e. through the *gateway*– commands locally to the devices based on events sensed by the *sensors*, generating added computational and memory load. In this case, it is defined by a scheme which specifies the rules and is validated by the agent responsible for the gateway.

4 An Organizational Model for the M2M Governance Strategy

Multi-agent organizations are concerned with the cooperation schemes between agents to achieve global goals [8] whether they result from agent interactions [17] or explicitly defined in terms of roles, plans, groups and links [8]. As M2M infrastructures should preferably be open to various applications and stakeholders, it is necessary to specify the governance strategies of such systems explicitly in order to guarantee that the requirements are fulfilled.

The $\mathcal{M}$OISE framework [10] provides a programming language to describe an organization following two independent dimensions: (i) the structural specification (SS) defines the roles and groups that the agents can adopt and enter in and (ii) the functional specification (FS) is a set of social schemes, i.e. a tree decomposition of goals organized into missions that the agents have to fulfil. The two dimensions are linked together by the normative specification (NS) assigning missions to roles. This makes $\mathcal{M}$OISE very suitable for the definition of a flexible governance strategy since we can envision to change either the SS, the FS or the NS without changing the other ones.

The organizational specification describes the governance objectives with respect to both *vertical* and *horizontal* concerns. In fact, as the infrastructure is shared by several heterogeneous applications and devices, the governance strategy must take into account the "horizontal" issues. Hence, a main frame defines the *horizontal* aspects which are extended for each *vertical* applicative requirements.

This section describes the organization corresponding to the M2M Organizational Specification (OS) for the governance strategy of the M2M architecture as described in Section 2 and highlights the key points for reorganization. This specification can be understood by the agents, so they can govern the M2M system based on it. Furthermore, they can reason about it to choose whether to follow it or not and then to adapt it to the situation.

4.1 Structural Specification

Fig. 4 shows the structural specification of the M2M OS. On one side it specifies the *horizontal* structure of an M2M architecture (within the M2M System group) and on the

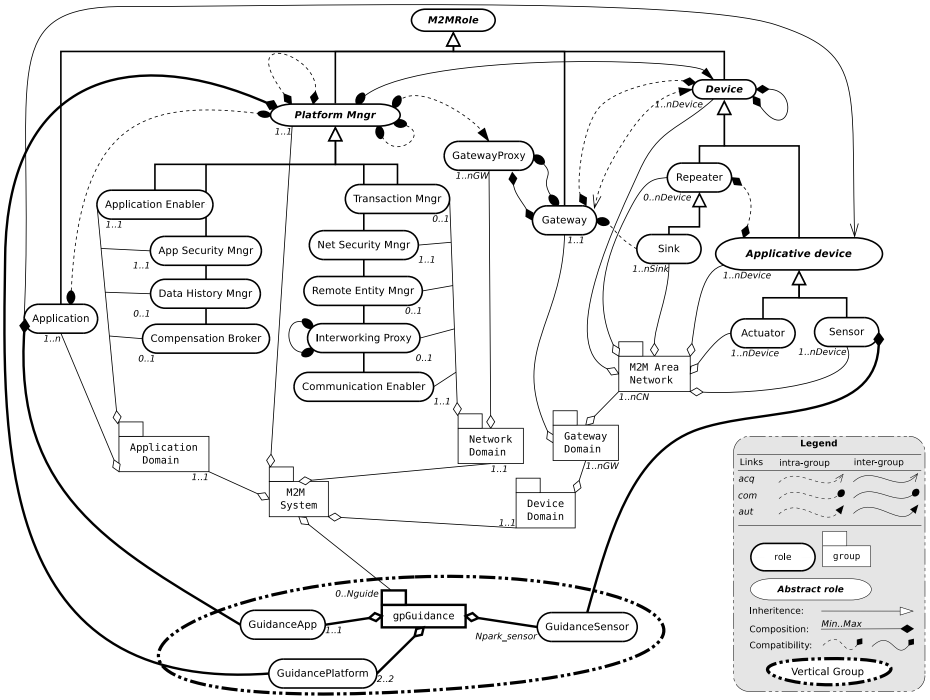

Fig. 4. Structural Specification for an M2M system and its extension for a "vertical" application (Car Guidance)

other side it defines one or several *vertical* groups (e.g. gpGuidance group). The agents can adopt one or several roles –depending on compatibility and cardinalities– to declare explicitly which part of the governed system they assume the responsibility of.

The *horizontal* part is composed of three groups corresponding to the M2M architecture described in Section 2.1: (i) the Device Domain group, (ii) the Network Domain group and (iii) the Application Domain group. This specification maps the functionalities of these different domains into roles.

The *Gateway* and *Gateway Proxy* roles are made compatible to allow agents to play both roles at the same time in the Device Domain and Network Domain groups respectively.

Each functionality of the M2M Core Platform is defined as a role in the Application Domain group –*Application Enabler*, *Application Security Manager*, *Data History Manager*, *Compensation Broker*– or in the Network Domain group –*Remote Entity Manager*, *Network Security Manager*, *Communication Enabler*, *Interworking Proxy*. All these roles inherit from the abstract *Platform Manager* role to make global the property of compatibility between any of these roles and to also express the communication link.

Finally, the *Application* role is responsible for performing the business logic by sending commands to the devices and by retrieving the collected data.

The *vertical* part is composed of specific groups which represent a contract between an application, the devices, the platforms and gateways. In particular, such a group expresses the application needs in terms of parts of the M2M system to be involved in the

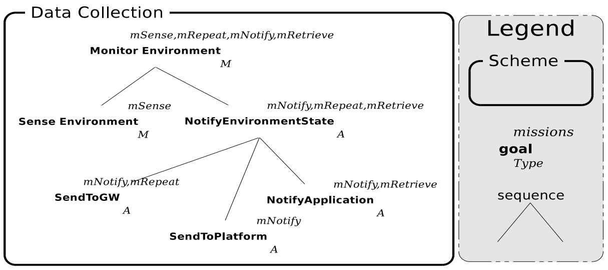

Fig. 5. Functional Specification: the Data Collection scheme

applicative realization and their cardinality. The adoption of a *vertical* role by the agents can be considered as a contract agreement. The content of this contract is defined by the norms which assigns missions to the roles specifying the requirements. The group is composed of roles linked to the *horizontal* specification using *compatibility* links. Thus, the agents can be involved in both the *vertical* and the *horizontal* organization to link the two concerns.

As an example, Fig. 4 shows such a *vertical* group for a *Parking Guidance* application (gpGuidance group). It involves the application itself (*GuidanceApp* role), the platform (*GuidancePlatform* role), a gateway (*GuidanceGW* role) and parking sensors (*Guidance-Sensor* role).

As shown in [11] this structure can evolve along different dimensions (roles, groups, cardinalities, links...). This allow the governance to suits to the infrastructure's dynamics. As an example, if more agents are needed to manage a part of the M2M system, the cardinality of the corresponding roles can be increased.

4.2 Functional Specification

The functional specification describes coordination schemes by means of goals to be globally satisfied by the agents in the organization. It gives the agents a comprehensive understanding of the system's functioning but it does not tell them how to achieve these goals: the agents are free to decide which actions to perform to satisfy the goals they are committed to. The following describes one of the social schemes that we have defined for *vertical* data collection.

The scheme in Fig. 5 describes the goals to collect data from sensors. The root goal (*Monitor Environment*), is a maintenance goal which is satisfied sequentially by (i) sensing the environment (*SenseEnvironment*) and (ii) notifying the subscribed applications the environment's state (*NotifyEnvironmentState*). The first sub-goal is accomplished in the context of the the *mSense* mission. The notification can be made either after each time something is sensed or after reporting several measures at once. It is an achievement goal satisfied by the following sub-goal sequence: (i) send to the gateway (*SendToGW*), (ii) send to the platform (*SendToPlatform*) and (iii) notify applications the sensed value (*NotifyApplication*).

Let's notice that the figure does not express the whole specification. In particular, each mission is qualified by the minimum and maximum number of agents to commit to; goals can be parameterized when instantiating the scheme. Agents can customize a scheme for a particular application or for special situations. For example, different areas of the city could be monitored differently depending on the traffic, the time of day or the user demand, then the agents can fine-tune these values to adapt the scheme to the situation in order to avoid unuseful message transmissions.

4.3 Normative Specification

Norms delimit the actions that are allowed in the system. In $\mathcal{M}$OISE a norm assigns a mission to a role, following a deontic relation when a condition is satisfied and specifies a finite time in which to fulfil it. Thus, it provides a flexible way of assigning tasks to the agents.

Table 2 summarizes a part of the norms used for the M2M system corresponding to the data collection scheme. Agents playing *Sensor* roles must sense the environment (*mSense* mission). This mission consists of sensing the environment either regularly (norm n_{01}), or at each change (norm n_{02}). In any case, Sensor agents must commit to the mSense mission (norm n_{03}).

Agents are free to decide whether to follow or violate these norms. It can be regulated by reinforcement or punishment to encourage them to follow the norms. But it also provides a way to detect irrelevant specifications: an agent might violate a norm because it is impossible to satisfy a goal. Then agents should either redefine the norm –e.g. modify the condition, relax the deontic relation– or the scheme itself –e.g. delete a goal or add an alternative to it; define a sequence to make the goal reachable.

Table 2. Normative specification for an M2M system

Id	Condition	Role	Rel.	Mission	TTF
n_{01}	$scheduled(sensing\_time)$	*Sensor*	perm	*mSense*	t_{sense}
n_{02}	$occurred(event)$	*Sensor*	perm	*mSense*	t_{sense}
n_{03}	$n_{01} \vee n_{02}$	*Sensor*	obl	*mSense*	t_{sense}
n_{04}	$changed(sensed\_value)$ $\wedge is\_critical(situation)$	*Sensor*	obl	*mNotify*	t_{send}
n_{05}	$t_{last\_msg} \geq msg\_period$ $\vee is\_full(buffer)$	*Sensor*	obl	*mNotify*	t_{send}
n_{06}	$on\_receive(msg)$	*Repeater*	obl	*mRepeat*	t_{repeat}
n_{07}	$on\_receive(msg)$ $\wedge is\_authenticated(msg)$	*Gateway*	perm	*mNotify*	t_{notify}
	... ...				

5 Application: Smart Parking Management with the SensCity Platform

This section describes the application of the governance model presented in Sections 3 and 4 to the scenario described in Section 2.2. It consists of an extension of the based organization by new roles, linked to the generic ones, grouped in a specific group that is responsible for specific schemes and ruled by specific norms.

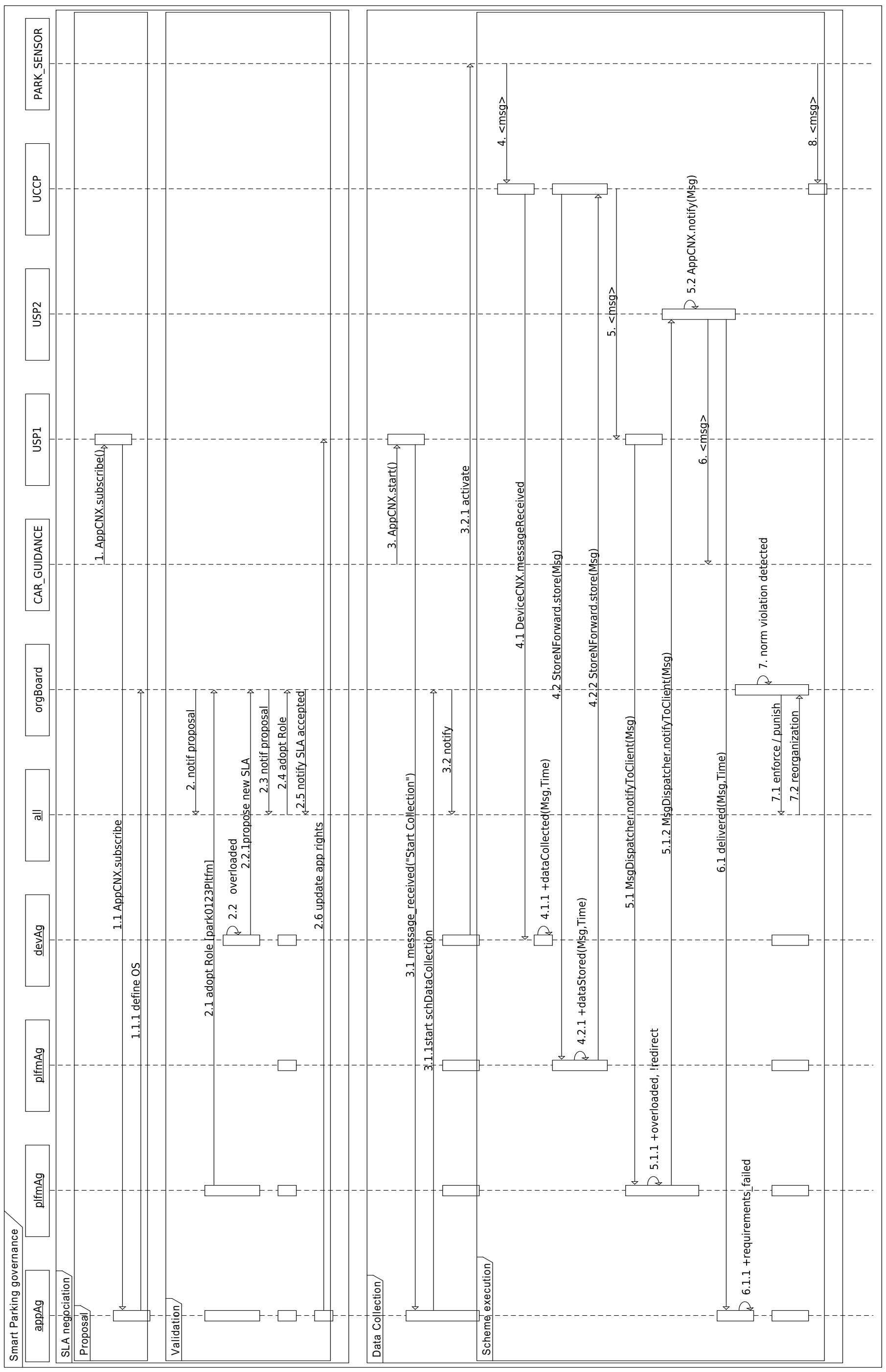

Fig. 6. Multi-agent governance of a smart parking management application

In the smart parking scenario the *CarGuidance* and *CityMonitoring* applications are governed by *appAg* agents while the *CarDetector*, *ParkingPost* devices are controlled by *deviceAg* agents. Each of the SensCity sub-platforms (*USP* and *UCCP* are controlled by a *platformAg* agent). In order to simplify and reduce the applicative description, no *Gateway* is considered here.

The following scenario is illustrated by the Fig. 6 sequence diagram in which the agents are the underlined participants, the *all* participant represents all the agents involved in the collaboration scheme, the *OrgBoard* artifact represents the organizational specification and controls, and *CAR_GUIDANCE*, *USP1*, *USP2*, *UCCP* and *PARK_SENSOR* represent elements of the M2M infrastructure, encapsulated by artifacts.

The first step consists of negotiating the SLA defined by a set of norms that manage the application-specific roles and missions. The SLA definition is done following the norm negociation cycle principle [3]: (i) a social scheme template is parameterized following the application needs, (ii) social norms define the SLAs for these goals and (iii) acceptance is achieved by role adoption. When the *CAR_GUIDANCE* application asks (1) the *USP1* platform for a subscription to a set of $N_{park\_sensors}$ devices, the *appAg* agent intercepts (1.1) the call and formulates the requirements as an organizational specification (1.1.1): a gpGuidance group composed of the *GuidanceApp*, *GuidancePlatform* (with cardinality 2) and *GuidanceSensor* (with cardinality $N_{park\_sensors}$) roles, a parameterized Data Collection scheme and norms corresponding to the application's requirements. All of the agents concerned are notified of the proposal (2). The agents can validate the SLA proposal by adopting one or several roles in the organization (2.1). But if they estimate that the proposal is not affordable, e.g. a resource will be overloaded (2.2), they can propose a new specification (2.2.1). This new proposal is communicated again (2.3) and the same process occurs until all of the agents validate the SLA by adopting the roles (2.4). Then the *appAg* will update the application's rights in the *USP1* platform.

Then the *CAR_GUIDANCE* application can start the subscription (3) which will be handled (3.1) by the *appAg* by starting the Data Collection scheme (3.1.1). The agents are notified of goals to achieve defined by the norms (3.2), so the *deviceAg* agents activate the *PARK_SENSOR* devices they are responsible for (3.2.1). When messages are received by the *UCCP* platform (4), the *deviceAg* makes sure that it meets the requirements (4.1, 4.1.1).

The *UCCP*'s *StoreNForward* component is encapsulated by an artifact, so a *platformAg* agent can regulate and validate the message's storage (4.2, 4.2.1, 4.2.2) before it is transmitted to the *USP1* platform (5). There, a *platformAg* agent interrupts the *MessageDispatcher* component (5.1) because it is overloaded (5.1.1) and decides to redirect it to the *USP2* platform (5.1.2) which transfers the message to the application (5.2, 6). This is notified to the *appAg* agent (6.1) which considers the requirements as not satisfied (6.1.1).

The norm violation (7) can be handled either by reinforcement and punishment mechanisms (7.1) applied to the agents and/or by a reorganization process (7.2) based on the analysis of the system's failures by the agents.

6 Related Work and Discussion

M2M is a promising paradigm and is the topic of several works. For instance, the SENSEI project[5] uses a virtual representation called "WSAN Islands" which provides the sensor value of the physical device and can be fed by predictive agents to reduce communications to sensors. Our work goes a step further since SENSEI doesn't yet provide any governance structure to control its components' behavior.

Some of them propose a governance structure. For instance, an agent-based approach is used by the US Ocean Observatories Initiatives to build an Ocean Observatories Initiative Cyberinfrastructure [5] to monitor the oceans with a marine sensor platform. It defines an infrastructure for an M2M server composed of six service networks interacting together following predefined interaction scenarios. The AAMSRT framework [14] gives another multi-agent approach for managing sensor re-targeting on satellites. Yet, both of these models use a static organizational model even if the second one is based on agents negotiating their commitment to missions.

Our paper proposes a template for an end-to-end M2M architecture. The openness and the heterogeneity of applicative requirements made necessary to ensure agility to the specifications by the means of organizational adaptations. For example, the *sensing_time* and t_{sense} values in norm n_{01} are specific to the applicative needs and the type of sensor. Agents can adapt the *OS* by extending the existing organization as a generic framework: new norms specific to applicative requirements, involving new roles extending existing ones to fulfil specific goals.

Moreover, as the *OS* is explicit and understandable for the agents, they can reason about it in order to improve the system's performances. For example, faced to a scalability issue, role cardinalities could be increased to enable some functionalities and components of the platform to be delegated to several agents. In contrast, security issues could lead to more atomicity by lowering role cardinalities.

Nevertheless, such a reorganization raises several issues: while self-organizing MAS's are more adaptive and robust, they might not converge in a stable organization [16]. Furthermore, the (re)organizational cost [12] must be taken into account to decide when to adapt. Another issue is to define who manages the reorganization: dedicated agents applying the organizational policy [11] or the applicative agents directly as in the AMAS [2] theory? While the former solution suffers robustness and scalability, the latter raises trust management issues. A possibility would be balancing between agents adapting locally based on their perception and dedicated agents for definitive organizational changes.

A contribution of our work is the explicitly expression of the *vertical* and *horizontal* concerns as different part of the organization. This model simplifies the analyze and reorganization process as the agent can clearly identify which part of the specification doesn't suit. Nevertheless, modifying a part of the organization raises issues with respect to the organization's integrity and consistency. For example, what does an agent is supposed to do when it is fulfilling a mission but the cardinality has just decreased in the *OS*? And what are the side effects?

[5] SENSEI (Integrating the Physical with the Digital World of the Network of the Future), EU ICT FP7, `http://www.sensei-project.eu/`

Hence, a solution is to stop the organization, update its definition and restart the groups and schemes. However, this approach is not satisfying neither as maintenance goals make difficult to define the time to stop a scheme. Furthermore, goals might have been modified so it is not possible to save all the schemes' states to restore them.

Therefore, a step further would be to split the model into several organizations: one for the *horizontal* concerns and one for each *vertical* application requirements. This would lead to several small organizations easier to analyze and adapt separately. Then agents will be able to reason in term of organization involvement in order to adapt the priority of one application's requirements or the horizontal objectives. Yet, much work have to be done to specify *extra-organization* links and constraints.

7 Conclusion and Further Work

Through a smart parking management application, this paper has presented a multi-agent architecture for an agile governance of Machine-to-Machine systems. The governance system is implemented using the JaCaMo framework in order to separate the different governance layers: CArtAgO artifacts are used to monitor and handle the M2M infrastructure, *Jason* agents applies the governance strategy and reason about the governance objectives which are specified as a $\mathcal{M}$OISE organization.

This governance model takes into account both the *vertical* application requirements and the shared infrastructure *horizontal* concerns, following the latest recommendations of the ETSI [6]. Furthermore, in order to meet scalability requirements, it highlights several key elements for adapting the structural, functional and normative specifications.

An other step in our work will focus on exploring reorganization aspects following two directions: (i) a behavioural specification to enable agents to adapt the organization directly and (ii) the definition of new roles dedicated to organization monitoring and reorganization processes to control the reorganization. Furthermore, while splitting the *horizontal* and the *vertical* concerns into several organization, it would be necessary to define *heuristics* to prioritize an organization over an other one.

In the meantime, the proposed organization, agents and artifacts will be deployed in an M2M infrastructure as a demonstrator in order to test and validate this model under real conditions.

References

1. Barkai, J.: Keynote speech – how to design your business (and products) for machine to machine communication. In: Connected World Conference (2008), `http://www.connectedworldmag.com/conference/index.php?q=2008-Presentations`
2. Bernon, C., Camps, V., Gleizes, M.P., Picard, G.: Engineering Self-Adaptive Multi-Agent Systems: the ADELFE Methodology, ch. 7, pp. 172–202. Idea Group Publishing (2005)
3. Boella, G., Van Der Torre, L.: Norm negociation in mutltiagent systems. International Journal of Cooperative Information Systems 16(1), 97–122 (2007)
4. Bordini, R., Hübner, J., Wooldridge, M.: Programming Multi-Agent Systems in AgentSpeak Using Jason. John Wiley & Sons, Ltd. (2007)

5. Chave, A., Arrott, M., Farcas, C., Farcas, E., Krueger, I., Meisinger, M., Orcutt, J., Vernon, F., Peach, C., Schofield, O., Kleinert, J.: Cyberinfrastructure for the US Ocean Observatories Initiative: Enabling interactive observation in the ocean. In: Oceans 2009-Europe, pp. 1–10 (May 2009)
6. ETSI: Tech. Spec. 102 690 V<0.13.3>, Machine-to-Machine communications – Functional architecture (July 2011)
7. Firesmith, D.: Profiling Systems Using the Defining Characteristics of Systems of Systems (SoS). Techreport, Software Engineer Institute, Pittsburgh, Pennsylvania (2010)
8. Horling, B., Lesser, V.: A survey of multi-agent organizational paradigms. The Knowledge Engineering Review 19(04), 281–316 (2005)
9. Hübner, J.F., Sichman, J.S., Boissier, O.: Developing Organised Multi-Agent Systems Using the MOISE+ Model: Programming Issues at the System and Agent Levels. Agent-Oriented Software Engineering 1(3/4), 370–395 (2007)
10. Hübner, J.F., Sichman, J.S., Boissier, O.: A Model for the Structural, Functional, and Deontic Specification of Organizations in Multiagent Systems. In: Bittencourt, G., Ramalho, G.L. (eds.) SBIA 2002. LNCS (LNAI), vol. 2507, pp. 118–128. Springer, Heidelberg (2002)
11. Hübner, J.F., Sichman, J.S., Boissier, O.: Using the MOISE+ for a Cooperative Framework of MAS Reorganisation. In: Bazzan, A.L.C., Labidi, S. (eds.) SBIA 2004. LNCS (LNAI), vol. 3171, pp. 506–515. Springer, Heidelberg (2004)
12. Kota, R., Gibbins, N., Jennings, N.: Decentralised Approaches for Self-Adaptation in Agent Organisations. ACM Transactions on Autonomous and Adaptive Systems, 1–36 (2011)
13. Lefauconnier, A., Gantelet, E.: Parking place search: strategies, associated nuisances, parking management issues in france. Tech. rep., SARECO, Paris, France (2005) (in French), `http://www.sareco.fr/Publications/Temps_de_recherche.pdf`
14. Levy, R., Chen, W., Lyell, M.: Software agent-based framework supporting autonomous and collaborative sensor utilization. In: Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009), pp. 93–100 (May 2009)
15. Ma, X., Luo, W.: The Analysis of 6LowPAN Technology. In: IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application, vol. 1, pp. 963–966. IEEE Com. Soc. (December 2008)
16. Picard, G., Hübner, J.F., Boissier, O., Gleizes, M.P.: Reorganisation and Self-organisation in Multi-Agent Systems. In: 1st International Workshop on Organizational Modeling, ORGMOD 2009, pp. 66–80 (June 2009)
17. Picard, G., Mellouli, S., Gleizes, M.-P.: Techniques for Multi-agent System Reorganization. In: Dikenelli, O., Gleizes, M.-P., Ricci, A. (eds.) ESAW 2005. LNCS (LNAI), vol. 3963, pp. 142–152. Springer, Heidelberg (2006)
18. Piunti, M., Boissier, O., Hübner, J.F., Ricci, A.: Embodied Organizations: a unifying perspective in programming Agents, Organizations and Environments. In: COIN10@MALLOW, pp. 98–114. Springer (September 2010)
19. Ricci, A., Piunti, M., Viroli, M., Omicini, A.: Environment programming in cartago. In: Bordini, R.H., Dastani, M., Dix, J., El Fallah-Seghrouchni, A. (eds.) Multi-Agent Programming: Languages, Platforms and Applications, vol. 2, pp. 259–288. Springer (2009)