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-ftgroup.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 city scale application, is faced to the increase in the number of services and applications that are deployed on it. It is thus necessary to dynamically share the infrastructure use between them. In this paper, we propose to use 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. network of smart devices –sensors and actuators– interacting with each other without human interventions. Recent improvements of low power wireless technologies [13] enable wireless devices to be connected to the Internet with a cheap deployment cost and with long term lifetime –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 systems, stakeholders are involved in different areas: application providers, constraint devices constructors, LLNs radio experts and telecommunication operators. However, the building of such vertical solutions at a city scale, are too expensive and not flexible enough. There is a growing need and interest for M2M infrastructures providing horizontal integration and sharing of devices between stakeholders.

This paper considers the practical use case issued from the SensCity[3] project which aims at providing a M2M platform for deploying multiple smart city services. This platform allows to connect heterogeneous wireless devices to a GPRS

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

gateway using Wavenis –a long range and energy efficient radio technology. From the application side, a standard access to the devices is provided via web services.

In order to be deployed at a city scale and used by different applications, such an infrastructure is faced to a multi-faceted problem of scalability [5] in terms of size, heterogeneity, topology, etc. To tackle with this problem an agile governance scheme is required to adapt to the different needs and requirements of the M2M applications deployed on such an infrastructure.

In this paper, we propose to use multi-agent technologies to define this governance layer on top of the M2M infrastructure. We have used a newborn multi-agent oriented programming framework named JaCaMo [4] to implement it. In order to get an agile governance, the governance strategy is defined as a normative organization, using the $\mathcal{MOISE}$ framework [7], part of the JaCaMo platform. Thanks to the explicit and agent-readable specifications, agents are able to reason on the governance strategy of the system to change and adapt it to the evolution of the system. The proposed multi-agent governance is illustrated with a smart parking management application.

In Section 2, we motivate our approach with the 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 the top of the M2M components (Section 3). We then focus, in sec. 4, on the definition of the governance strategy and how it can be dynamically adapted by the 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 compare it to related works. Finally, Section 7 concludes this paper and draws the perspectives of future works.

2 Motivations

Machine to Machine (M2M) systems are an early technology which is just getting out of full proprietary solutions with different standard proposals [4]. We first give an overview of the *M2M architecture* standard proposal on which we found our work. We then introduce the smart parking management application as an illustrative example where we highlight the need of an agile governance layer.

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 [4], 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

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

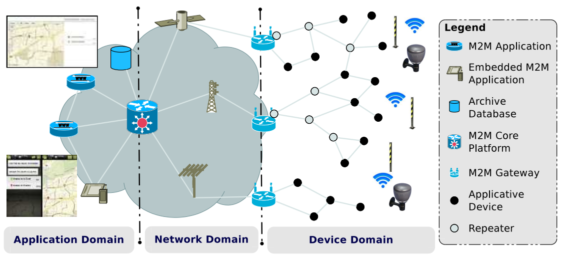

Fig. 1: An end-to-end M2M architecture [4].

(WSAN) linked to a gateway. The WSAN groups several devices communicating together. Devices can embed several sensors and actuators, but also none of them: *repeaters* are placed to extend the coverage managed by a *gateway*. It manages one or several WSAN, security and devices authentication and can also manage quick reaction to sensed events generating commands to devices. The *gateway* sends/receives messages to/from the *platform* via a broadband access. Thus it belongs also to the Network Domain. The *core platform* is involved in both Network Domain and Application Domain, as summarized in Table 1. On one side, it is responsible for network communications and interaction with other platforms. On the other side, it gives the *application* –managing the business logic– a transparent access to the devices and stores the messages. Readers seeking for further details might refer to the *ETSI M2M Functional Specification* [4].

Table 1: M2M core platform's functionalities [4].

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

As the infrastructure is shared by heterogeneous applications, agility is required for managing different requirements. Considering a Smart Parking system, car detectors are used for monitoring parking places. When a car parks or leave the place, the event is notified through the M2M infrastructure: a message is (1) sent to the gateway which (2) authenticates and (3) forwards it to the M2M platform where (4a) it is stored and (4b) notified to subscriber applications, which in turn (5) retrieve the message. Applications can also send commands to the devices using the reverse path to act on the environment, eg. to raise parking post to reserve a place. In this scenario, let us consider the two following applications involving different requirements: *Drivers Guidance* and *City Monitoring*.

On one hand, the *Drivers Guidance* application helps drivers to find a parking place directly and close to their destination following its preferences, reducing traffic flow and pollution[5]. 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.

On the other hand, the *City Monitoring* application is used by city services (eg. police) to monitor non stationary places. It requires alerts to be sent when a place is occupied during a non stationary time with a variable priority (eg. firemen water places has priority over garage doors).

As sending messages consumes a lot of energy, several applications sharing the same devices with heterogeneous requirements, it raises several issues such as scalability and energy consumption. Further more, other applications, using other devices, will share the same infrastructure –the platform and the gateways– generating traffic and ressources management issues. In this context, an agile governance is required to define how the resources –devices, servers, network– should be used. 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 governance of M2M systems. To clearly separate the different concerns that arise in such applications, we have chosen the JaCaMo [6] [16] platform. 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* [2], *(ii)* $\mathcal{M}$OISE [7], and *(iii)* CArtAgO[17]. A JaCaMo multi-agent system (i.e., a software system programmed in JaCaMo) is given by an agent organization programmed in $\mathcal{M}$OISE, 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.

These three dimensions are used to define the governance layer deployed on top of the M2M infrastructure aiming at governing its use by the different ap-

[5] 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 [11] (2005).

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

plications: (*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 take local decisions with respect to the governance taking into accounts their partial view on the infrastructure status and that cooperate with the other agents participating to 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 and consuming the data provided by the M2M infrastructure. The organization limits the space 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 and decide to change the organization when this one is no more adapted to the current state of the governance requirements (eg. high number of violations greater than authorized by the contract).

Before defining the governance strategy as an organization in the next section (Section 4), this section describes the use of artifacts (Section 3.1) to monitor and to control the *SensCity* platform and the different agents (Section 3.2) of the proposed governance model.

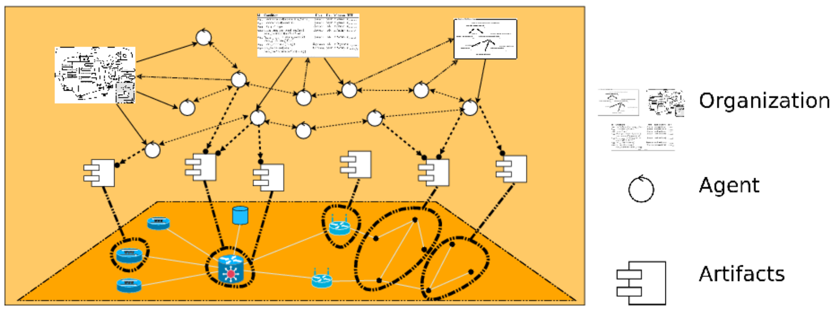

Fig. 2: *Agents* manage the M2M infrastructure using *Artifacts* following the specification defined by the *Organization*.

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 MAS the control of it. In the context, of the *SensCity* project, the governance layer is deployed on top of two platforms: the *Urban Service Platform (USP)* to manage the notifications to the applications, their rights and billing, and the *Urban Collect and Command Platform (UCCP)* to manage the devices and communications with them. Figure 3 describes the component-based architecture of these two platforms. Artifacts encapsulate the components' functionality.

These artifacts are used to give to the agents a representation of the system to govern. On one hand, an artifact monitors one or several components' activity: status and calls to the components are notified to the agents by signals and observable properties. On the other hand, the artifact's operations enable the agents to use the component.

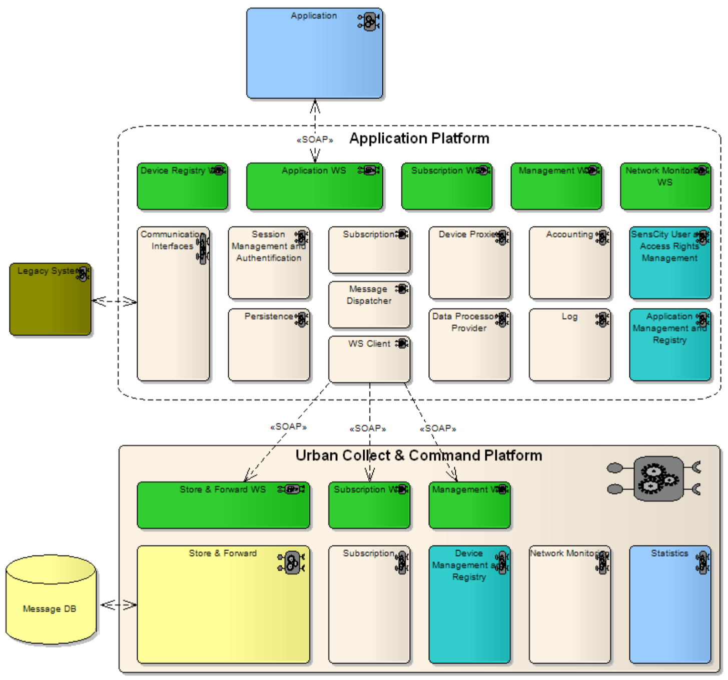

Fig. 3: The *USP* and *UCCP* platforms' components are encapsulated by CArtAgO artifacts to monitor them and give control to the agents.

3.2 Agents to apply and reason on the governance strategy

Agents are the decision taking entities of the governance layer. They adopt one or several roles in the organization corresponding to the part of the governance they assume the responsibility of. Following the strategy specification given by organization (described in Section 4), they ensure 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.
From their experience, they can reason on the strategy definition and evaluate

it with respect to the M2M system's functioning. Indeed, some of the specifications might be inapplicable or possible to improve. They can, then, redefine the strategy by proposing new organizational specifications.

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

The *appAg* agents control the commands and requests sent by the application to the devices to check the compliance with the specification. An *AppCNXArt* artifact encapsulate the *ApplicationCNX* component of the platform which allow an *appAg* agent to intercept the messages so that the application does not violate the requirements. In the mean time, it controls the messages going to the applications which allow it to check that application's requirement are also satisfied. This allow the *appAg* agents to evaluate the relevance of the requirements, in order to propose to the other agents to enforce or relax them.

Similarly, the *deviceAg* agents controls the usage of the devices specified by the contract, by using encapsulation artifacts such as *DeviceCNXArt*, to ensure 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 a smarter usage of the device (eg. 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 the priority to one contract over an other one.

The *platformAg* agents are responsible for the platform's 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 heavy, 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 too be done accordingly to the priority of the message and its destination: priority to notifications to the *PoliceMonitoring* application the over *CarGuidance* ones.

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 locally –ie. by the *gateway*– commands 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 validated by the agent responsible for the gateway.

The applicative requirements are specified using an organization. A group defines a contract between an application, the devices, the platforms and gateways. The group is composed of roles inherited from the based. The adoption of these roles by the agents can be considered as a contract agreement. The content of this contract, defined by the norms, assign missions to the roles specifying the requirements thanks to the parameterized goals and norms. Such a specification is defined between each of the applications and each set of devices involved together. The next section describes the organization corresponding the M2M governance strategy and the key point for reorganization.

4 An organizational model for the M2M governance strategy

Multi-agent organizations are concerned with the cooperation schemes between agents to achieve global goals [6] whether they result from agents interactions –self-organizing MAS [15]– or they are explicitly defined in terms of roles, plans, groups and links –organization centered MAS [6]. As M2M infrastructures are wanted to be open to various applications and stakeholders, it is needed to specify explicitly the functioning of such systems in order to guarantee the requirements fulfillment.

The $\mathcal{MOISE}$ framework [8] describes an organization following two independent dimensions: (i) the structural specification (SS) defines the roles and groups which the agents can commit to and (ii) the functional specification (FS) is a set social schemes, ie. a tree decomposition of goals, organized in missions, the MAS have to fulfill. The two dimensions are linked by the normative specification (NS) assigning roles to missions. This makes $\mathcal{MOISE}$ very suitable for the definition of flexible governance models. The remainder of this section describes an Organizational Specification (OS) for the governance strategy of the M2M architecture as described in Section 2. This specification is understandable by the agents so they can govern the M2M system based on it. Further more, they can reason on it to choose whether to follow it or not and then to adapt it to the situation.

4.1 Structural specification

Figure 4 shows an agent organization structured into 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 describes the functionalities of the different parts of the system in terms of roles. The agents can adopt one or several roles –following the compatibility and cardinalities– to explicitly declare which responsibility they assume to govern. This way, it gives to the agents a semantic to the different part of the system.

The Device Domain group is composed of roles for making M2M devices grouped in a *M2M Area Network* communicating to each other and for managing these devices by gateways in their *Gateway Domain* group. Sub roles of the *Device* abstract role are compatible with each other, even with *Device* roles in other groups, but can communicate only inside the same group. *Applicative Device* roles –*Sensor* and *Actuator*– are responsible for managing the execution of applicative commands, while the *Repeater* role involves the agent in forwarding messages in the sensor network and the *Sink* role to forward messages from/to the gateway.

The *Gateway* and *Gateway Proxy* roles are compatible and are responsible for managing the gateway tasks in, respectively, the Device Domain and Network Domain groups.

Each functionality of the M2M Core Platform is defined as a role in the Application Domain group –*Application Enabler*, *Application Security Manager*, *Data*

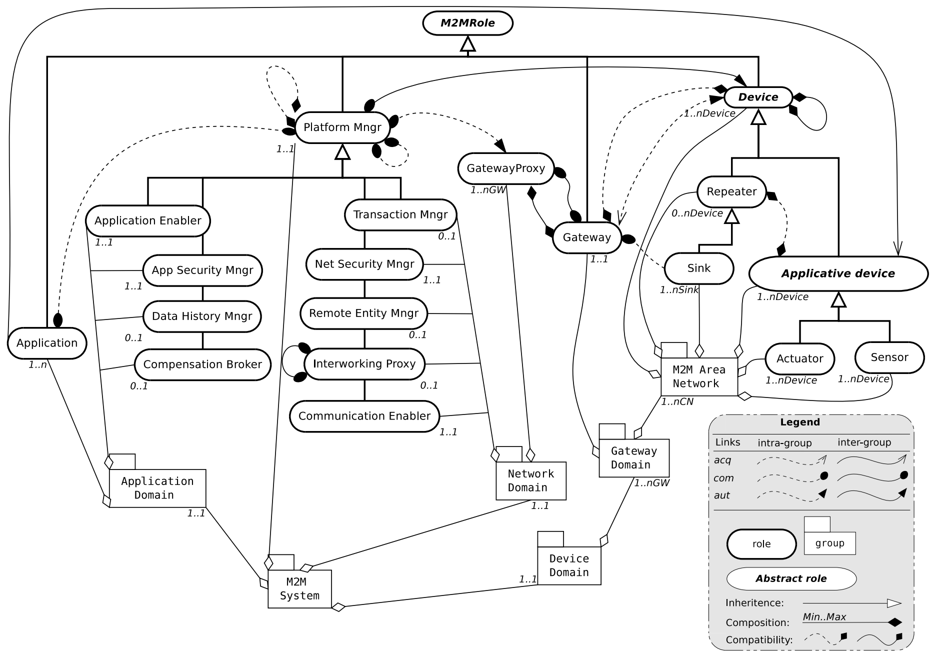

Fig. 4: Structural Specification for a M2M system.

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 *Platform Manager* role, and so, are compatible and communicate together.

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

The roles corresponding to the shared infrastructure are detailed following the ETSI specifications, but the *Application*, *Sensor* and *Actuator* roles can be extended by inheritance to explicitly specify the application. For example, the *Drivers Guidance* application (resp. the *City Monitoring* application) will involve the *GuidApp*, *GuidSensor*, *GuidReserv* roles (resp. *CityApp*, *CitySensor*). Further more, this structure can evolve by redefining roles cardinality and compatibility. As an example, the platform can be duplicated to balance the load of the server, therefore more agents could decide to assume the same role –ie. functionality– together.

4.2 Functional specification

The functional specification describes coordination schemes by means of goals to be globally satisfied by all the agents of the organization. It gives to the agents a comprehensive understanding of the system's functioning but it does not tell them how to achieve these goals, and so, they are free to decide which actions to perform to satisfy the goals. The following describes one of the social schemes.

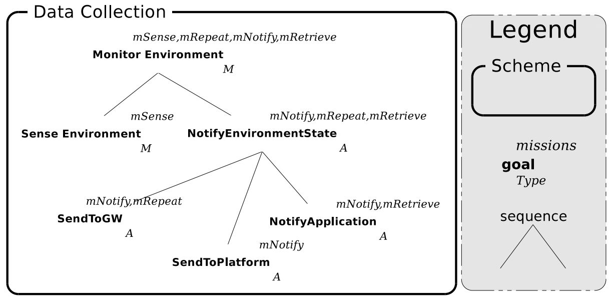

Fig. 5: Functional Specification: the Data Collection scheme.

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

Such a figure does not express the whole specification, in particular each mission defines the minimum and maximum number of agents to commit to and goals can parameterized when instantiating the scheme. This enables the agents to customize a scheme for a particular application or to special situations. For example, different areas of the city could be monitored differently depending on the traffic, the time of the day or the user demand, then the agents can tune these values to adapt the scheme to the situation in order to save messages transmission.

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 matched and specifies a finite time to fulfil it. Thus, it gives 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 in 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}).

Table 2: Normative specification for a 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}
		... ...			

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 gives 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 –eg. modify the condition, relax the deontic relation– or the scheme itself –eg. delete a goal or add an alternative to it, define a sequence to make the goal reachable.

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 previous organization, linked to it by new roles inherited from the generic ones grouped in a specific group responsible for specific schemes and ruled by specific norms.

In the smart parking scenario the *CarGuidance* and *PoliceMonitoring* 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* is 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 Figure 6 sequence diagram in which the agents are the underlined participants, the *all* participant represent all the agents involved 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 one or several artifacts.

The first step consists in negotiating the SLA; which consists in a set of norms that manage the application-specific roles and missions. When the *CAR_GUIDE* 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 formulate the requirements as an organizational specification (1.1.1): a gpGuide group composed of the *GuideApp*, *GuidePlatform* (with cardinality 2) and *GuideSensor* (with cardinality $N_{park\_sensors}$) roles, a parameterized Data Collection scheme and norms corresponding to the application's requirements. The proposal is notified to all the concerned agents (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, eg. a resource will be overloaded (2.2), they can propose a new specification (2.2.1). This new proposal is notified again (2.3) and the same process occurs until all 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_GUIDE* 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 of (3.2.1). When messages are received by the *UCCP* platform (4), the *deviceAg* controls that it is suiting 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 interrupt 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 are 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 failures by the agents.

6 Related work and Discussion

M2M is a promising paradigm and is the topic of several work. The SENSEI project[7] uses a virtual representation called "WSAN Islands" which provide the sensor value of the physical device and can be fed by predictive agents to reduce communications to sensors. Yet it does not provide any governance structure to control its components' behavior.

An agent-based approach is used by the US Ocean Observatories Initiatives to build an Ocean Observatories Initiative Cyberinfrastructure [3] to monitor the oceans with 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 [12] gives another multi-agent approach for managing sensor re-targeting on satellites. Both of these models

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

use a static organizational model even if the second one is based on agents negotiating their commitment to missions.

The given organization provides a governance template for an end-to-end M2M architecture. However, considering the openness and the heterogeneity of applicative requirements, such an organization needs agility and so should be adaptable. For example, the *sensing_time* and t_{sense} values in norm n_{01} are specific to the applicative needs and the type of sensors. Agents can adapt the *OS* by extending the existing organization as a generic framework: new norms specific to applicative requirements, involving new roles inherited from 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. Facing to scalability issue, role cardinalities could be increased, to enable some functionalities and components of the platform to 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 are more adaptive and robust, they might not converge to a stable organization [14]. Furthermore, the (re)organizational cost [10] 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 [9] or the applicative agents directly as in AMAS [1] 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 organization changes.

7 Conclusion and further work

Through a smart parking management application, the current paper has presented a multi-agent architecture for an agile governance of Machine-to-Machine systems, following the latest recommendations of the ETSI [4]. However, in order to suit to scalability requirements, it highlights several keys for adapting the structural, the functional and the normative specifications.

The next step of our work will focus on exploring these reorganization aspects following two directions: (i) a behavioral specification to enable agents to adapt directly the organization and (ii) the definition of new roles dedicated to the organization monitoring and reorganization processes to control the reorganization. In the mean time, the proposed organization, agents and artifacts will be deployed in an M2M infrastructure as a demonstrator in order to test and validate this model in real conditions.

References

1. Bernon, C., Camps, V., Gleizes, M.P., Picard, G.: Engineering Self-Adaptive Multi-Agent Systems : the ADELFE Methodology, chap. 7, pp. 172–202. Idea Group Publishing (2005)

2. Bordini, R., Hübner, J., Wooldridge, M.: Programming Multi-Agent Systems in AgentSpeak Using Jason. John Wiley & Sons, Ltd (2007)
3. 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. Oceans 2009-Europe pp. 1–10 (May 2009)
4. ETSI: Tech. Spec. 102 690 V<0.13.3>, Machine-to-Machine communications – Functional architecture (Jul 2011)
5. Firesmith, D.: Profiling Systems Using the Defining Characteristics of Systems of Systems (SoS) (2010)
6. Horling, B., Lesser, V.: A survey of multi-agent organizational paradigms. The Knowledge Engineering Review 19(04), 281–316 (Nov 2005)
7. 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)
8. Hübner, J.F., Sichman, J.S., Boissier, O.: A Model for the Structural, Functional, and Deontic Specification of Organizations in Multiagent Systems. In: 16th Brazilian Symposium on Artificial Intelligence. vol. 2507, pp. 118–128. Springer (2002)
9. Hübner, J.F., Sichman, J.S., Boissier, O.: Using the MOISE+ for a cooperative framework of MAS reorganisation. In: 17th Brazilian Symposium on Artificial Intelligence (SBIA'04). pp. 506–515. Springer (2004)
10. Kota, R., Gibbins, N., Jennings, N.: Decentralised Approaches for Self-Adaptation in Agent Organisations. ACM Transactions on Autonomous and Adaptive Systems pp. 1–36 (2011)
11. Lefauconnier, A., Gantelet, E.: La recherche d'une place de stationnement : strategies, nuisances associees, enjeux pour la gestion du stationnement en france
12. Levy, R., Chen, W., Lyell, M.: Software agent-based framework supporting autonomous and collaborative sensor utilization. In: Autonomous Agents and Multi-Agent Systems (2009)
13. 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 (Dec 2008)
14. 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, ORGMOD09. pp. 66–80 (Jun 2009)
15. Picard, G., Mellouli, S., Gleizes, M.P.: Techniques for Multi-agent System Reorganization. In: Engineering Societies in the Agents World VI. pp. 142 – 152. Springer Link (2006)
16. 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 (Sep 2010)
17. 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. Springer (2009)

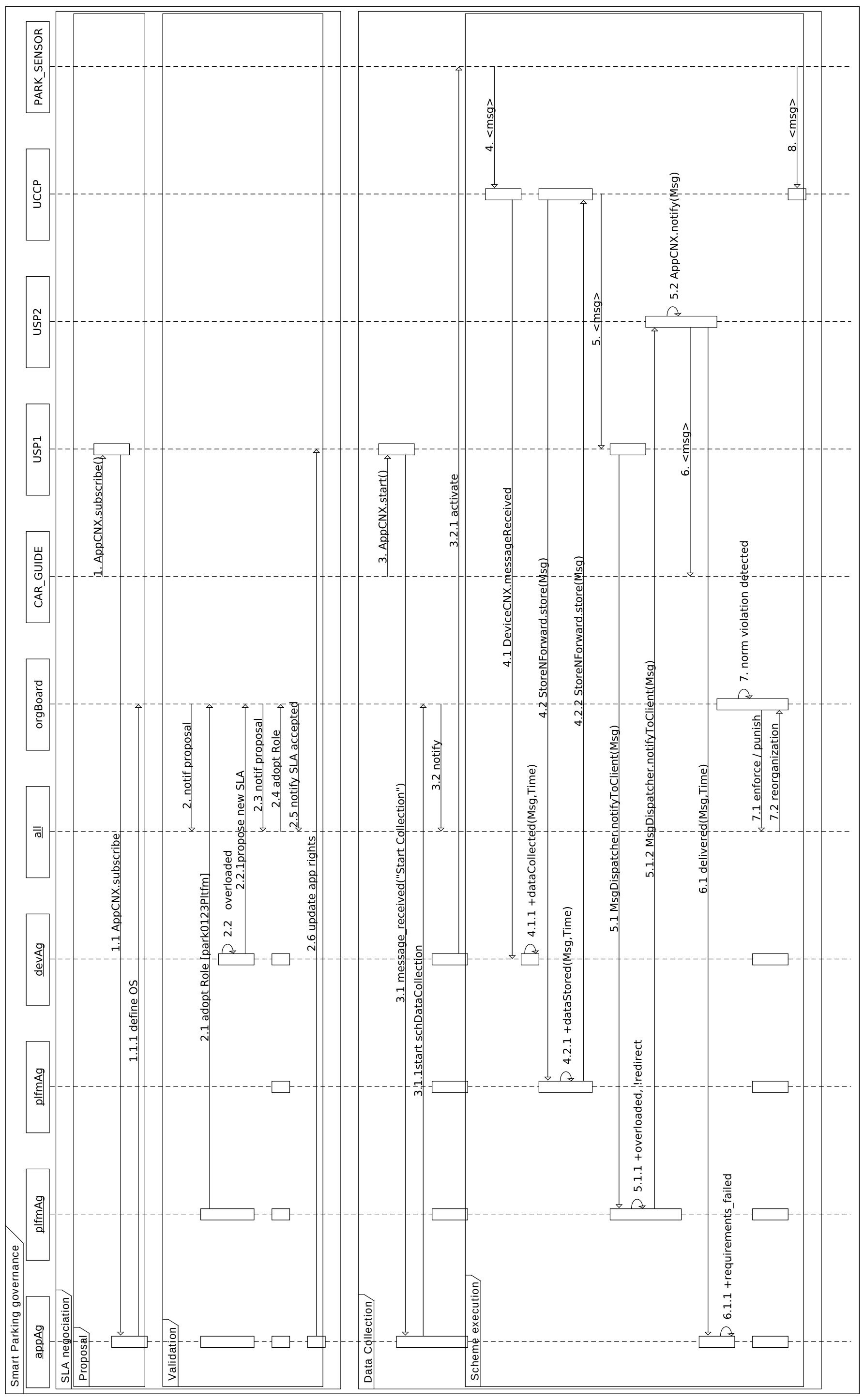

Fig. 6: Multi-agent governance of a smart parking management application.