Organisation multi-agent pour la gouvernance de systèmes *Machine-to-Machine*

C. Persson[a,b] persson@emse.fr
G. Picard[b] picard@emse.fr
F. Ramparany[a] fano.ramparany@orange-ftgroup.com
O. Boissier[b] boissier@emse.fr

[a] Orange Labs Network and Carrier, TECH/MATIS, Grenoble, France

[b] LSTI/Institut Henri Fayol, Ecole Nationale Supérieure des Mines de Saint-Etienne, France

Résumé

Le paradigme Machine-to-Machine *(M2M) implique des appareils (capteurs, effecteurs) interagissant pour fournir des services localisés dans le monde physique. Avec la maturité du M2M, émerge une demande grandissante pour des solutions mutualisées dans lesquelles les applications peuvent partager un ensemble commun d'appareils. Dans ce contexte, le projet* SensCity *propose une infrastructure pour mettre en œuvre des applications à l'échelle de la ville, ce qui nécessite de fournir des moyens de gouvernance agile pour prendre en compte l'*extensibilité *du système (ie.* scalability*). Nous proposons d'utiliser les technologies multi-agents pour répondre à cette problématique. Selon cette approche, la stratégie de gouvernance est exprimée par une organisation multi-agent à l'aide du framework organisationnel MOISE. Nous illustrons notre proposition par un système de gestion intelligente du stationnement.*

Mots-clés : *M2M, SMA, modèle organisationnel, ville intelligente*

Abstract

The Machine-to-Machine (M2M) *paradigm involves devices (sensors, actuators) interacting together to provide services located in the physical world. As the technology is gaining maturity, there is a growing need for mutualized solutions in which applications can share a common set of devices. In this context, the* SensCity *project proposes an infrastructure to enable mutualized city scale applications. Implementing such an infrastructure raises the problem of providing an agile governance with respect to scalability. Besides, Multi-Agent technologies grant adaptability, flexibility and proactivity properties to such a decentralized applications. Thus, this paper proposes a Multi-Agent organization for expressing the governance strategy of such systems. Using the MOISE framework, we illustrate how it is used within a smart parking management application.*

Keywords: *M2M, MAS, Organizational model, Smart City*

1 Introduction

La prochaine génération de ville –les *villes intelligentes* ou *smart cities*– fournira des services automatisés pour améliorer la vie de leurs citoyens (e.g. ramassage optimisé des déchets ménagés, télérelève de compteurs, adaptation du trafic, gestion du stationnement). Ces services à valeur ajoutée font un usage intensif d'objets communicants [1] –capteurs ou effecteurs– interagissant sans intervention humaine : on parle de systèmes *Machine-to-Machine* ou *M2M*. En effet, les progrès récents des technologies sans-fil à faible consommation [12] permettent aux devices sans-fil d'être connectés à l'Internet pour un faible coût de déploiement et pour une durée de vie assez longue (20 ans). Cependant, les applications M2M nécessitent de nombreux acteurs métier : fournisseurs d'applications, fabricants de devices, experts radio et opérateurs de télécommunications. Ainsi, les solutions "verticales" (i.e. chaîne complète de l'application aux devices dédiée à une seule application) sont trop coûteuses et rigides pour l'échelle de la ville. Par ailleurs, différents acteurs expriment un besoin grandissant pour une intégration "horizontale" et un partage de compétences et de devices est exprimé par les différents acteurs.

Cet article considère un cas d'étude concret issu du projet SensCity [2] qui a pour but de fournir une plateforme M2M mutualisée pour déployer des services dans une ville. Cette plateforme permet de connecter des devices sans fil très hétérogènes à une passerelle GPRS utilisant Wavenis –une technologie radio efficace en énergie et à longue portée. Du point de vue applicatif, elle fournit un accès standardisé aux devices. Afin d'être déployé dans des villes, un tel système doit considérer différentes dimensions telles que sa *taille*, son *hétérogénéité*, sa *topologie* et son immersion dans le *monde physique* ; toutes ces dimensions contribuent au problème de passage à l'échelle dans les appli-

1. Dans le reste de l'article, nous parlerons de *devices*.
2. FUI Minalogic : http ://senscity.minalogic.net/http ://senscity.minalogic.net/

cations M2M [14]. Une telle architecture nécessite un schéma de gouvernance agile de telle sorte qu'elle puisse évoluer sans problème de passage à l'échelle. De plus, il est très difficile de définir tous les besoins des applications M2M *a priori* à cause de leurs grandes hétérogénéité et ouverture. Dans ce contexte, les *systèmes multi-agents* (ou *SMA*) offrent des modèles et des technologies pour concevoir des systèmes distribués et complexes [8]. De plus, les SMA permettent de concevoir des organisations ouvertes avec des agents hétérogènes basés sur des technologies disparates [5]. L'objectif de cet article est de proposer un modèle de gouvernance d'infrastructure M2M par un SMA. Ainsi, organisation multi-agent y est définie en utilisant le framework *MOISE* [6]. Grâce aux spécifications explicites et interprétables par les agents, ces derniers pourront raisonner sur la structure de gouvernance afin de se ré-organiser et de la changer de manière agile. Cette organisation SMA est illustrée au travers d'une application de gestion intelligente du stationnement urbain.

Après avoir donné une description de l'architecture M2M et de l'application de gestion du stationnement en section 2, un modèle organisationnel est présenté en section 3 et une représentation agent des composants M2M est décrite en section 4. Ensuite, la section 5 discute cette approche ainsi que des travaux connexes. Enfin, la section 6 conclut cet article et dresse quelques perspectives.

2 Contexte et applications du M2M

Cette section présente une vue d'ensemble de l'architecture M2M et introduit le scénario d'application de gestion du stationnement, puis, discute des besoins d'agilité dans la gouvernance de tels systèmes pour faire face au problème de passage à l'échelle.

2.1 Architecture *Machine-to-Machine*

Le M2M est une technologie encore à ses débuts qui sort tout juste des solutions propriétaires. Cependant, le comité technique M2M de l'ETSI travaille sur les standards pour ces infrastructures. Cet article est basé sur la dernière version des spécifications de l'ETSI [3] [4]. qui propose une architecture divisée en trois domaines (cf. Fig. 1) : device, réseau et application. Dans cette architecture les passerelles (*gateways*) et la plateforme centrale font le lien entre deux domaines.

3. European Telecommunications Standards Institute

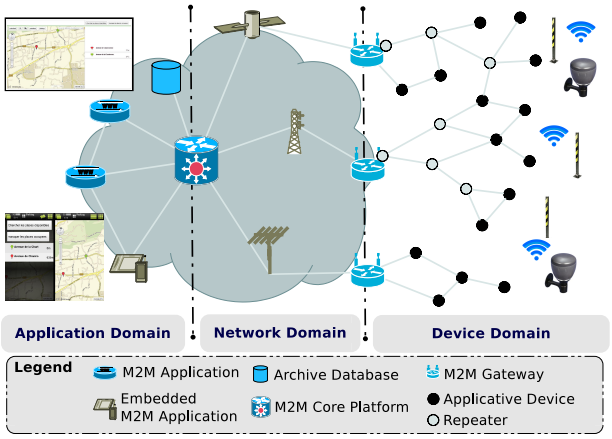

FIGURE 1 – Architecture M2M de l'ETSI [4].

Le domaine des devices est composé de devices applicatifs –*capteurs* et *effecteurs*– et éventuellement de *répéteurs* communicant dans un réseau sans fil (WSAN) connecté à la *passerelle*. Un WSAN regroupe plusieurs *devices* communicant entre eux. Ces devices peuvent embarquer zéro ou plusieurs capteurs et effecteurs. Des répéteurs sont sont placés pour étendre la portée d'un réseau relié à une *passerelle*.

La *passerelle* gère un ou plusieurs WSAN, ainsi que la sécurité et l'authentification des devices. Elle peut aussi générer des commandes aux *effecteurs* afin de réagir plus rapidement aux événements perçus. La *passerelle* envoie et reçoit les messages transmis entre les *devices* et la *plateforme M2M* via un accès large bande. Elle est donc située entre les domaines des devices et celui du réseau.

La *plateforme M2M* se trouve à la jonction des domaines du réseau et des applications. La table 1 résume les différentes fonctionalités de la *plateforme* selon les domaines. D'un côté, elle gère les communications ainsi que les interactions avec d'autres *plateformes*, et de l'autre, elle fournit aux *applications* –qui gèrent les logiques métiers– un aspect transparent aux *devices*.

TABLE 1 – Fonctionnalités de la plateforme M2M définies par l'ETSI [4]

Domaine du Réseau	Domaine des Applications
Remote Entity Management (**REM**	Application Enablement (**AE**)
Generic Communication (**GC**)	Compensation Brokerage (**CB**)
Reachability Addressing and Repository (**RAR**)	Transaction Management (**TM**)
Communication Selection (**CS**)	History Data Retention (**HDR**)
Interworking Proxy (**IP**)	
⟵ Security (**SEC**) ⟶	

2.2 Gestion intelligente du stationnement

Le projet SensCity vise à partager les infrastructures et les réseaux existants entre différentes applications M2M. Le scénario présenté dans cette section illustre un exemple d'applications déployées dans cette infrastructure.

Une application de gestion intelligente du stationnement est un système M2M qui surveille l'occupation des places de parking pour guider les conducteurs et ainsi contribuer à la réduction du trafic et de la pollution. En effet, la recherche de stationnement est estimée responsable de 5% à 10% du trafic et représente une perte de 70 millions d'heures pour un coût de 600 millions € en France [10] (chiffre 2005). Ce système peut également être utilisé pour surveiller des places non prévues pour le stationnement (accès aux bornes d'incendie, sorties de garage, etc...). Pour ce faire, des détecteurs de voitures sont disposés sous les places. Comme l'envoi de messages consomme beaucoup d'énergie, les capteurs pourront alerter le système des changements ou les enregistrer afin de les envoyer regroupés en messages statistiques, en fonction du temps, du jour, ou de sa position. Les messages sont envoyés à la passerelle GPRS grâce à des répéteurs. Toutes les données sont collectées par un serveur –la plateforme M2M– qui notifie les applications : GPS des conducteurs, services de planification urbaines de la ville, et commissariats. De plus, afin de garantir au conducteur de trouver un stationnement libre, les futurs systèmes de gestion de stationnement permettront également aux conducteurs de le réserver depuis leur véhicule en activant un plot télescopique. Le conducteur pourra recevoir un e-ticket qui sera utilisé pour redescendre le plot à son arrivée.

Dans ce scénario, trois applications sont impliquées, avec, chacune, différents besoins : (1) la réservation et cartographie des stationnements pour le guidage des conducteurs, (2) la planification urbaine et (3) la surveillance policière. Les services de planification urbaine et de police se soucient de l'occupation des places en différents termes : la planification nécessite uniquement des statistiques sur toutes les places pour planifier l'aménagement urbain en stationnement, alors que la police a besoin d'être alertée lorsque des stationnements interdits sont occupés. Les applications de cartographie et de réservation fournissent une vue de la disponibilité des places aux clients et envoient les commandes de réservation aux effecteurs. Lorsqu'un utilisateur authentifié souhaite réserver, l'application cherche plusieurs places de stationnement et notifie le client avec un e-ticket de réservation s'il en trouve. Le e-ticket a une durée limitée (15 minutes) pour désactiver le plot et indiquer la position exacte de la place réservée.

2.3 Gouvernance agile pour le M2M

Comme les systèmes M2M sont déployés dans le monde physique, les interventions sur les devices (ie. déploiement ou maintenance) sont très coûteuses. Ainsi, partager les ressources entre plusieurs acteurs permet de diminuer ce coût et d'offrir la possibilité de construire des applications plus complexes. Ceci mène au besoin d'assurer l'extensibilité [4] (*scalability*) du système en le dotant d'un mode de gouvernance agile respectueux de ses propriétés essentielles. L'extensibilité est abordée selon des dimensions de passage à l'échelle, i.e. des métriques pour évaluer le système suivant ses caractéristiques [14]. Ces dimensions peuvent être regroupées en quatre catégories –la *taille* (e.g. nombre de places de parking surveillées), l'*hétérogénéité* (e.g. gestion du stationnement couplée avec un système de transport multi-modal), la *topologie* (e.g. densité du réseau de capteurs) et l'immersion dans le *monde réel* (e.g. dynamique de la ville)– suivant trois types de propriétés –*utilisation* (e.g. latence, disponibilité), *performance* (e.g. efficacité, robustesse) et *administration* (e.g. coût).

Ces dimensions ont un impact dans tous les domaines du système : du nombre et des types de devices déployés dans un WSAN à la couverture géographique couverte par les applications (e.g. bâtiment/ville/région/...). Dans ce contexte, notre but est de proposer, dans un premier temps, une organisation multi-agent pour gouverner les systèmes M2M afin de les doter ultérieurement de capacité de réorganisation et d'auto-organisation pour plus d'agilité.

3 Modèle organisationnel M2M

Cette section, et la suivante, décrivent l'approche multi-agent utilisée pour définir la gouvernance d'applications M2M. Nous avons utilisé la plateforme JaCaMo [17] (Jason [2] + CArtAgO [18] + $\mathcal{MOISE}$ [6]) afin de séparer clairement les différentes préoccupations qui se posent dans de telles applications : les *agents* pour les différentes entités de prise de décision, les *artefacts* pour les ressources accessibles et partagées entre les agents et l'*organisation* pour

4. Aptitude d'un produit ou d'un système à fonctionner correctement, sans perdre ses propriétés essentielles, lors d'un changement d'échelle d'un ou plusieurs paramètres. *Journal Officiel*, 23/02/2003

exprimer l'architecture de gouvernance régulant le fonctionnement des différents agents autonomes. JaCaMo fournit un cadre unifié pour le développement de ces trois aspects du SMA.

Les organisations multi-agents concernent les schémas de coopérations entre agents pour atteindre des buts globaux [5]. Ces schémas peuvent résulter du fonctionnement des agents dans le cas de SMA auto-organisateurs [16]. Ils peuvent être explicites en termes de rôles, de plans, de groupes et de liens dans le cas de SMA centrés organisation [5]. Comme les infrastructures M2M sont souhaitée ouvertes à des applications et acteurs variés, il est nécessaire de spécifier explicitement le fonctionnement de tels systèmes afin de garantir le respect des exigences globales. Dans cette section, une spécification organisationnelle est proposée pour la gouvernance d'architectures M2M vues en section 2.

$\mathcal{MOISE}$ [6] définit une organisation suivant deux dimensions : (i) la spécification structurelle (SS) et (ii) la spécification fonctionnelle (FS) ; reliées ensemble par la spécification normative (NS). Ainsi $\mathcal{MOISE}$ est particulièrement adapté à la définition de modèles de gouvernance flexibles.

La *spécification structurelle* (section 3.1) est un ensemble de *groupes* composés de *rôles*. En fonction de cette spécification, les agents s'engagent dans un rôle en fonction de leurs compétences et créent des groupes suivant la spécification organisationnelle. Un rôle définit une place dans l'organisation régie par différent type de liens avec les autres rôles : accointance, communication, autorité et compatibilité. Quand un agent s'engage dans un rôle, il s'engage à satisfaire les normes concernant les plans pour atteindre les buts collectifs du système.

La *spécification fonctionnelle* (section 3.2) définit un ensemble de *buts* pour l'organisation et un ensemble de *misssions* à remplir pour atteindre ces buts, organisés en *schémas sociaux*. Un schéma social est un *plan* pour atteindre un but, décomposé en un arbre de *sous-buts*, décrits en utilisant un opérateur de plan et une liste de sous-buts. Deux types de buts peuvent être définis : les buts ponctuels ou *achievement* (A), devant être atteints une seule fois, et les buts de *maintenance* (M) devant être constamment satisfaits. Chaque but a un temps maximum pour être accompli ou *time-to-fulfill* (TTF). Pour chaque schéma, un ensemble de missions regroupent les buts du schéma en des ensembles cohérents pouvant être alloués aux agents. Comme les rôles, les missions ont une cardinalité qui définit le nombre minimum/maximum d'agents pouvant y souscrire.

Enfin, la *spécification normative* (section 3.3) assigne les missions aux rôles. Les normes définissent comment les missions peuvent ou doivent être achevées et par quels rôles. Une norme n spécifie une relation déontique d entre un rôle r et une mission m avant un temps TTF lorsque la condition c devient vraie.

3.1 Spécification structurelle

Les rôles sont utilisés pour attribuer aux agents des parties de l'infrastructure M2M à surveiller et à piloter afin d'en assurer le bon fonctionnement. Suivant la description donnée en section 2.1, l'organisation M2M est structurée en trois groupes (voir figure 2) : (i) le groupe Device Domain, (ii) le Network Domain et (iii) l'Application Domain.

Le groupe Device Domain est composé de rôles pour regrouper les devices communiquant ensemble –ie. dans une même zone, indépendamment de leur logique métier– dans un M2M Area Network et pour gérer ces devices par passerelle dans leur Gateway Domain. Chaque rôle a une définition et une cardinalité dans l'organisation. L'engagement des agents est validé *si et seulement si* la cardinalité et les contraintes de compatibilité entre rôles sont respectées. Voici la liste des rôles d'une infrastructure M2M :

- *Gateway* [1..1] : responsabilité de l'authentification des devices et des mises-à-jours logicielles et peut assumer des fonctions applicatives comme l'envoi de commandes vers les devices pour réagir rapidement à certains événements perçus par les capteurs. Ainsi, ce rôle a un lien d'autorité sur le rôle *Device* et un lien de communication avec le *Gateway Proxy*.
- *Device* [1..$N_{devices}$] : abstrait qui peut communiquer avec les autres *Devices* du même M2M Area. Ils ont tous une relation d'accointance avec *Gateway* qui est responsable du réseau de proximité même s'il ne peut communiquer directement avec lui. Il est possible pour un agent de s'engager sur des rôles *Device* dans différents groupes M2M Network Area. Dans certains cas, les *Device*s peuvent communiquer directement avec la *Platform*.
- *Repeater* [0..$N_{devices}$] : relais des messages envoyés par d'autres devices afin d'atteindre la *Gateway*. Tout *Device* peut avoir à jouer ce rôle afin de gérer les surcoûts entraînés par les communications pour l'appareil.
- *Sink* [1..N_{sink}] : même responsabilité que le rôle *Repeater* mais il peut communiquer directement avec la *Gateway*.
- *Applicative Device* [1..$N_{devices}$] : rôle abstrait, responsable de l'exécution des commandes et de la satisfaction des requêtes applicatives. Il

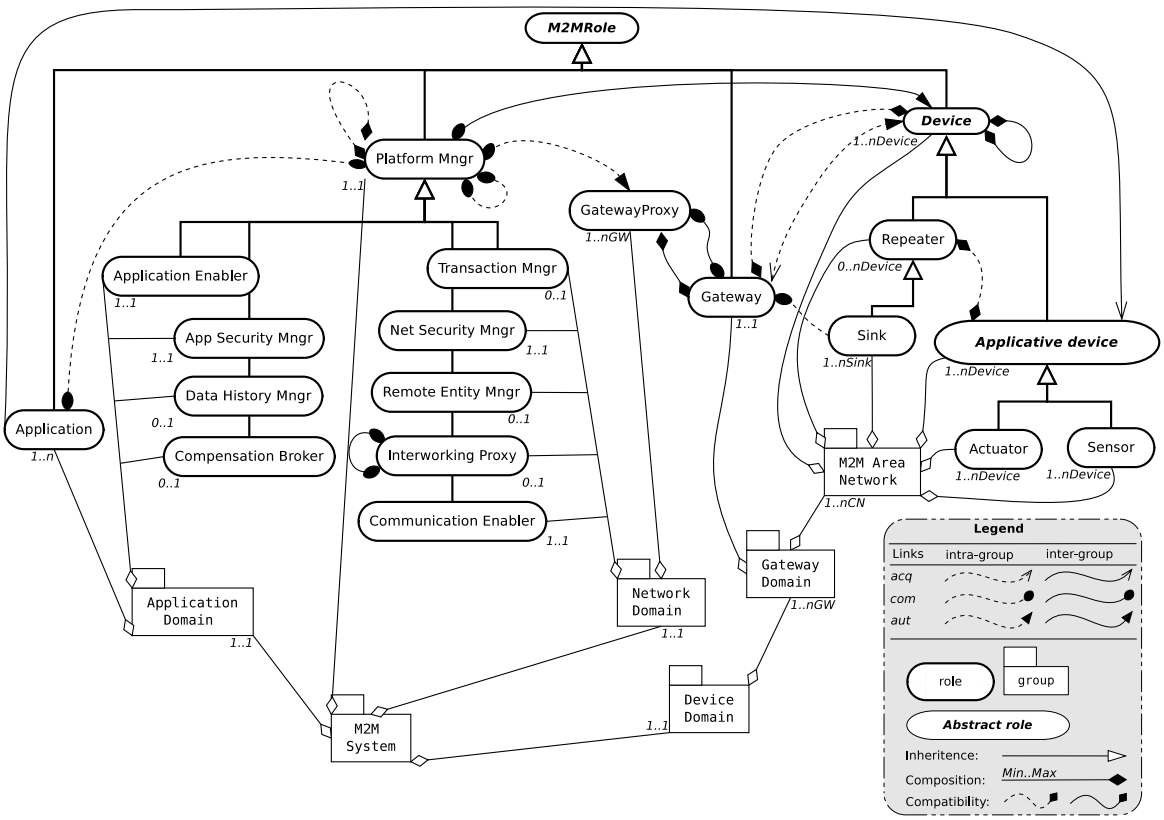

FIGURE 2 – Spécification structurelle pour un système M2M.

est compatible avec le rôle de *Repeater*. Ce rôle est différencié sémantiquement par les sous-rôles *Actuator* et *Sensor*.
– *Actuator* $[0..N_{devices}]$: hérite du rôle *Applicative Device*, responsable des actions agissant sur l'environnement. Ces actions peuvent être planifiées ou ordonnées à la demande par commande.
– *Sensor* $[0..N_{devices}]$: hérite du rôle *Applicative Device*, responsable des mesures d'un aspect de l'environnement et de l'envoi des notifications. Les messages peuvent être planifiés ou provoqués par un état critique.

Comme la plateforme centrale M2M est impliquée à la fois dans le domaine réseau et le domaine des applications, un rôle neutre –dont les rôles de plateformes hériteront dans le groupe Network Domain ou le groupe Application Domain– est défini comme suit :

– *Platform Manager* [1..1] : responsable de la gestion du *M2M Core Platform*. Les rôles héritant de celui-ci sont compatibles entre eux et peuvent communiquer avec les autres sous-rôles dans la même plateforme. Il peut communiquer avec les *Application*s et a autorité sur tous les *Device*s.

Les rôles héritant du rôle *Platform Manager* correspondent aux différentes fonctionnalités présentées en section 2.1. Le groupe Network Domain est composé de différents rôles responsable de la gestion et de l'interaction avec les devices via une passerelle. Certains rôles encapsulent des composants de la plateforme centrale. Les rôles de ce groupe sont :

– *Gateway Proxy* $[N_{GW}..N_{GW}]$: responsable de la communication avec la plateforme et de la lier avec le Device Domain. Il peut communiquer avec un seul *Gateway*. Ces deux rôles sont également compatibles et peuvent être joués par le même agent *GatewayAg*.
– *Remote Entity Manager* [1..1] : gestion de la configuration des passerelles et des devices (mises-à-jour logicielles) (voir **REM**).
– *Network Security Manager* [1..1] : gestion de l'authentification des devices et des passerelles (voir **SEC**).
– *Transaction Manager* [1..1] : gestion de l'exécution des actions qui nécessitent plusieurs devices (voir **TM**).
– *Communication Enabler* [1..1] : responsable du routage des messages aux devices en utilisant le chemin le plus pertinent. Ainsi, il gère également l'accessibilité et la mobilité des devices (voir **GC**, **CS** et **RAR**).
– *Interworking Proxy* [1..1] : fourni une communication transparente avec d'autres plateformes M2M (voir **IP**).

Enfin, le groupe Application Domain inclut tous les rôles correspondant aux services fournis par la plateforme pour permettre aux applications d'accéder aux devices applicatifs :

- *Application* [1..$N_{application}$] : fournit un service utilisant des données de capteurs ou contrôlant des effecteurs. Pour ce faire, il peut communiquer avec le *Platform Manager* pour manipuler les *Sensor*s et *Actuators*.
- *Application Enabler* [1..1] : interface pour les *applications* pour répertorier les services M2M fournis, l'accès aux ressources et la réception des notifications (voir **AE**).
- *Application Security Manager* [1..1] : vérification de l'authenticité des *applications* et validation de l'accès aux ressources (voir **SEC**).
- *Data History Manager* [0..1] : archivage des transactions pertinentes suivant des profils d'applications ou de devices, sans interprétation du message (voir **HDR**).
- *Compensation Broker* [1..1] : gestion du courtage entre les applications clientes et les ressources. Responsable de la gestion des compensations financières, des conflits et priorités d'accès aux devices –particulièrement pour les effecteurs (voir **CB**).

Cette spécification étant explicite et compréhensible par les agents, ils peuvent raisonner sur la structure de l'organisation et la modifier : augmenter/diminuer des cardinalités de rôles, renforcer des liens de communication ou créer de nouveaux rôles. Ils peuvent également ajouter des liens de compatibilité en fonction de leurs capacités, en considérant un rôle comme un sous-ensemble des capacités de l'agent.

3.2 Spécification fonctionnelle

Cette section définit la spécification fonctionnelle qui gouverne les buts et missions du système M2M, à travers l'exemple de deux schémas : *collecte de données* et *commande de device*.

Collecte des données (Fig. 3a). Consiste à rapporter les données perçues par les capteurs aux applications. Le but racine (*MonitorEnvironment*) est un but de maintenance qui est satisfait par la séquence (i) percevoir l'environnement (*SenseEnvironment*) puis (ii) notifier de l'état de l'environnement (*NotifyEnvironmentState*). Le premier sous-but peut consister à être informé de tout changement ou de percevoir régulièrement une valeur. Il est accompli par la mission *mSense*. De la même manière, la notification peut être faite dès la perception d'un changement ou être une synthèse de plusieurs données perçues. La notification est un but ponctuel satisfait par la séquence suivante : (i) envoyer à la passerelle (*SendToGW*), (ii) envoyer à la plateforme (*SendToPlatform*) et (iii) notifier les applications (*NotifyApplication*).

Commande de devices (Fig. 3b). Consiste à envoyer une commande aux devices. Cette commande peut être un simple ordre ou une planification à l'avance d'actions. Pour atteindre le but racine (*CommandDevice*), il est nécessaire de mettre en œuvre la séquence suivante : (i) envoyer la commande (*SendCmd*), (ii) effectuer la commande (*PerformCmd*) et (iii) notifier du succès de la commande (*NotifyCmd*).

Les buts de ces schémas sociaux sont regroupés dans les missions suivantes :

- *mSense* [1..N_{sensor}] pour percevoir l'environnement, régulièrement ou occasionnellement ;
- *mExec* [1..$N_{actuator}$] pour exécuter la commande ;
- *mNotify* [3..N_{sensor} + N_{GW} + $N_{platform}$] pour prévenir les entités M2M de changements dans l'environnement ;
- *mRepeat* [0..N_{hops}] pour relayer les messages à leur destination dans le WSAN (les exigences de *latence* limitent le nombre de sauts à $N_{hops} \leq 5$) ;
- *mCmd* [3..N_{app} + N_{GW} + $N_{platform}$] pour élaborer et transmettre une commande aux devices ;
- *mRetrieve* [1..N_{app}] pour collecter des données que la plateforme a produit grâce aux devices ou à propose des devices ;
- *mArchive* [0..1] pour archiver les messages transitant au travers de la plateforme.

Ces missions regroupent des buts de manière cohérentes tels qu'ils soient assignés en même temps aux agents. De plus, un but peut être compris dans plusieurs missions et être ainsi réalisés par différents rôles dans des conditions différentes. Par exemple, la figure 3b montre que le but *Send to GW* fait parti des missions *mNotify* et *mRepeat*.

Concernant le scénario de gestion de parcs de stationnements, le schéma *Data Collection* est effectué pour connaître le taux d'occupation des places de parking et le schéma *Commande Device* est utilisé pour réserver ou libérer une place de parking. Les détecteurs de voitures perçoivent chaque voiture dès qu'elle se gare ou qu'elle libère une place. Mais pour économiser de l'énergie, il peut être préférable de ne pas envoyer trop de messages de notification. De même, il n'est pas nécessaire de connaître le

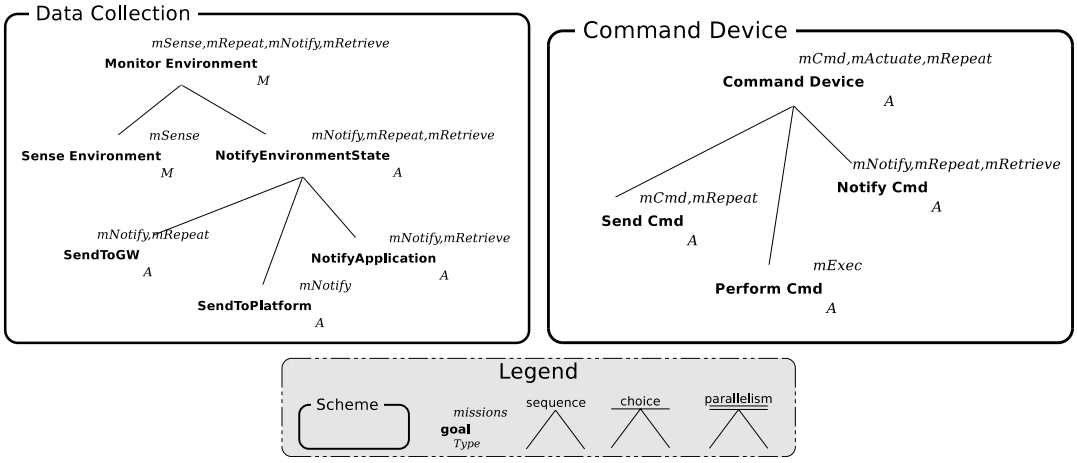

FIGURE 3 – Spécifications fonctionnelles pour le M2M : collecte de données (a) et activation (b)

nombre exacte de place libres s'il y en a beaucoup ou qu'il n'y a pas de demande. Ainsi, les valeurs *TTF* peuvent être adaptées en fonction de l'heure, du jour, de l'emplacement ou même du type d'application.

Dans la perspective d'un modèle de gouvernance agile, les agents devraient être capables d'adapter ces exigences en fonction de la situation et de leur propre exigences. Ceci signifie que les agents ont besoin de détecter des buts non atteignables ou trop coûteux. Alors, ils pourraient décider et proposer de nouvelle définitions de schémas sociaux : redéfinir les *TTF* ou les cardinalités, changer l'ordre des buts, voire même créer de nouveaux sous-buts pour compenser des buts non atteignables.

3.3 Spécification normative

Cette section définit les normes qui contraignent les actions que les agents peuvent effectuer dans le système M2M. La table 2 résume certaines de ces normes relatives aux schémas présentés précédemment : les normes n_{01} à n_{07} définissant comment les données doivent être perçues et envoyées à la plateforme, alors que les normes n_{10} à n_{16} spécifient les conditions pour envoyer des commandes aux effecteurs.

Les agents jouant le rôle de *Sensor* sont supposés percevoir l'environnement, en effectuant la mission *mSense*. Comme précisé précédemment, cette mission peut consister à percevoir l'environnement régulièrement (n_{01}) ou à la demande (n_{02}). Dans tous les cas, les agents *Sensor*s doivent s'engager dans la mission *mSense* (n_{03}). Cependant, la valeur t_{sense} peut varier dans les deux cas pour des raisons métiers. Par exemple, ceci peut être vu comme un compromis résultant de la négociation entre des applications pour préserver et économiser les capteurs.

Les commandes sont générées par les agents *Application* (n_{10}). Une commande est transmise par le *Application Enabler si et seulement si* l'*Application* a les droits sur le device en question (n_{11}). Les rôles de *Gateway* et *Repeater* sont impliqués dans la transmission de la commande à l'agent *Actuator* (n_{06}, n_{12}, n_{13}). Ainsi, un agent *Actuator* effectue l'action (n_{14}) et notifie du succès de cette dernière (n_{15}, n_{16}).

Ces normes sont directement applicables au cas de la gestion de stationnement : réservation et libération de places. Mais, dans le deuxième cas, l'état du succès n'est pas si important, comme le système détectera la place comme étant occupée. Par conséquent, il devrait être possible de changer la relation déontique de la norme n_{18} en *permission*. Une autre possiblité est de diviser cette norme en deux et d'ajouter une condition sur le succès de l'action (monter ou descendre la borne).

TABLE 2 – Spécifications normative pour le M2M

Id	Condition	Rôle	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}
n_{10}	–	Application	perm	*mCmd*	t_{cmd}
n_{11}	$is\_auth(app) \wedge has\_right(app, dev)$	App Enabler	obl	*mSend*	t_{send}
n_{12}	$on\_receive(cmd)$	Gateway	perm	*mSend*	t_{send}
n_{13}	$on\_receive(cmd) \wedge high\_priority(msg)$	Gateway	obl	*mSend*	t_{send}
n_{14}	$on\_receive(cmd)$	Actuator	obl	*mExec*	$t_{actuate}$
n_{15}	$succeeded(cmd)$	Actuator	obl	*mNotify*	t_{Cnotif}
n_{16}	$failed(cmd)$	Actuator	obl	*mNotify*	t_{Cnotif}

4 Système multi-agent pour le M2M

Dans cette section, une représentation agentifiée de l'architecture M2M présentée en section 2 est donnée. L'objectif principal des agents est de s'assurer du bon fonctionnement des éléments de l'infrastructure M2M. La figure 4 illustre les différentes entités M2M gérées par différents types d'agents : *DeviceAg*, *GatewayAg*, *PlatformAg* et *AppAg*.

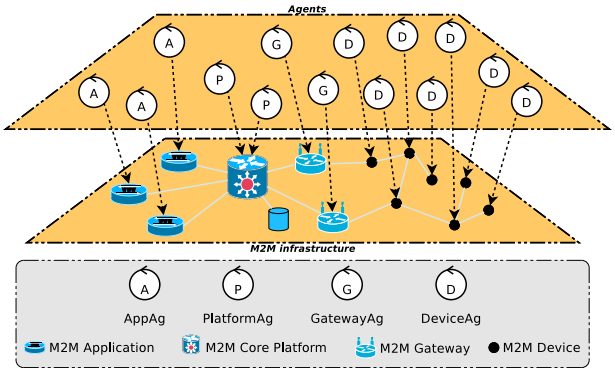

FIGURE 4 – Les *agents* gèrent l'infrastructure M2M par le biais d'*artefacts*.

Un agent *DeviceAg* pilote un device M2M qui interagit avec le monde physique pour réaliser certaines fonctions. Chaque *DeviceAg* gère un *device* : il s'assure de son bon fonctionnement en contrôlant la quantité des messages émis, leur fréquence et la latence. Les *devices* étant fortement contraints, l'agent gère les différentes les fonctionnalités engagées dans le reste du système en adoptant les rôles *Sensor*, *Actuator* et *Repeater* et en engageant le *Device* dans différentes missions. Ainsi, par exemple, le même *Device* peut être *Sensor* dans un schéma et *Repeater* pour un autre.

Un agent *AppAg* représente une *application* cliente utilisant des *devices*. Il adopte donc le rôle *Application* dans l'organisation. Il sélectionne les *devices* selon ses besoins et envoie les commandes et requêtes à ces derniers, en négociant les accès avec les agents *DeviceAg*.

Ces deux types d'agents utilisent des artefacts *DeviceArt* comme représentation des *devices*. L'agent *DeviceAg* accède à l'état physique du *device* –alimenté par les messages de statuts et/ou inféré en fonction des données statistique– et accorde les droits aux agents *AppAg*. Ces derniers lisent ses `capacités` afin de construire les commandes dont ils ont besoin et de les envoyer, l'arrivée de message leur est signalée par un signal. Ils peuvent ensuite le récupérer via la *plateforme M2M*.

La *plateforme* est gérée par un agent *PlatformAg* qui évalue sa charge afin d'en assurer un bon fonctionnement. Il authentifie les messages entrant, les transmets à leurs destinataire –ie. *applications* et *devices* via la *passerelle*– et les stocke si nécessaire. En cas de dysfonctionnements, il peut déléguer une partie des traitements à d'autres agents *PlatformAg* pour répartir la charge. Il assume pour cela les rôles *Platform Manager*, *Application Enabler*, *Data History Manager*, *Application Security Manager*, *Network Security Manager*, *Compensation Broker*, *Communication Enabler*, *Remote Entity Manager*, *Transaction Manager* et *Interworking Proxy*.

Un agent *GatewayAg* pilote une *passerelle* en assumant les rôles *Gateway* et *GatewayProxy*. Il assure la transmission des messages entre les *devices* et la *plateforme* en optimisant les communications en fonction de la priorité des messages. Il peut aussi générer des commandes en réponses aux événements perçus par les *capteurs* en fonctions de règles simples communiquées par les *applications*.

La *plateforme* et les *passerelles* du projet SensCity reposent sur des serveurs dont l'architecture est à base de composants (Java Beans). Chaque composant implémente une des fonctionnalités présentées en section 2.1 et est encapsulé par un artefact correspondant. Cet artefact fournit un accès naturel au composant pour les agents en fournissant des opérations correspondant aux méthodes du composant. Il est ainsi facile pour les agents d'utiliser mais aussi de manipuler (eg. dupliquer) ces composants.

Les agents *PlatformAg* et *GatewayAg* sont définis indépendamment des applications métiers, alors que les agents de types *AppAg* et *DeviceAg* sont spécifiques aux politiques de leur propriétaires –ie. respectivement l'application *cliente* et le *fournisseur* de device. Dans le scénario de gestion de parc de stationnement *CarGuidanceNReservAg*, *UrbanPlanAg* et *PoliceMonitorAg* sont des agents *AppAg* alors que *CarDetectorAg* et *ParkPostAg* sont des agents *DeviceAg*.

5 Travaux connexes et discussion

Le M2M est un paradigme prometteur et est le sujet de nombreux travaux. Le projet FP7 SENSEI [5] propose une architecture virtuelle pour regrouper des réseaux de capteurs dans des "WSAN Islands". Chaque capteur se voit affecter un capteur virtuel qui fournit les données

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

provenant du capteur physique. Le projet fournit une infrastructure décentralisée et flexible mais reste focalisé sur les technologies et n'offre aucun modèle de gouvernance pour spécifier les comportements des composants et de gestion du niveau applicatif.

Une approche orientée agent est utilisée par le US Ocean Observatories Initiatives pour développer la Ocean Observatories Initiative Cyberinfrastructure pour surveiller les océans avec une plateforme de capteurs marins [3]. Ce projet définit l'infrastructure pour un serveur M2M composé de 6 réseaux de services interagissant suivant des scénarios prédéfinis. Le framework AAMSRT propose une autre approche multi-agent pour la gestion de capteurs embarqués sur des satellites [11]. Ces deux approches utilisent un modèle organisationnel statique même si le second est basé sur des agents négotiant pour s'engager dans des missions.

Les sections précédentes ont décrit un modèle organisationnel pour la gouvernance de systèmes M2M afin de garantir un fonctionnement global grâce à des règles sur l'adoption de rôles et les interactions entre les agents. Cependant, considérant le caractère ouvert et hétérogène de l'infrastructure M2M, il apparaît qu'une telle organisation devrait être capable d'évoluer afin de gérer agilement les situations non planifiées. Par exemple, les valeurs *sensing_time* et t_{sense} de la norme n_{01} sont spécifiques aux besoins applicatifs. Ainsi l'agilité pourrait venir des agents eux-même par l'extension de l'organisation considérée en tant que cadre générique : de nouvelles normes spécifiques aux besoins applicatifs, faisant intervenir des rôles spécifiques hérités des rôles existant pour réaliser des schémas sociaux paramétrés selon les besoins, peuvent être le résultat d'une négociation entre les agents *AppAg* et *DevAg*.

Comme les *spécifications organisationnelles* sont explicites et compréhensibles par les agents, ils peuvent raisonner dessus pour améliorer les performances du système. Dans le cas du problème de passage à l'échelle, les cardinalités des rôles peuvent être augmentées pour déléguer des tâches à d'autres agents. À l'inverse, des problèmes sécurité peuvent conduire à plus d'atomicité en réduisant la cardinalité maximale d'un rôle.

Cependant de telles réorganisations soulèvent plusieurs questions [15] : *Quand ? Quoi ? Qui ?*

Quand changer l'organisation ? Décider quand adapter l'organisation est important d'après deux aspect du système : la stabilité et la performance. En effet, il est nécessaire de prendre en compte le coût de la réorganisation [9], surtout dans le cas de systèmes très contraints comme le M2M. De plus, si les systèmes émergents et auto-organisateurs sont plus adaptatifs et robustes, ils peuvent difficilement garantir de converger vers une organisation stable [19]. Mais la réorganisation peut être nécessaire pour réagir aux événements imprévus (e.g. nouvelles applications, ressource indisponible) ou une perte de performance (e.g. surcharge du trafic). Les systèmes M2M ont besoin d'un support à long terme qui signifie que de nouveaux rôles pourraient apparaître dans l'organisation et certaines contraintes pourraient disparaître alors que de nouvelles pourraient apparaître. Ainsi, le modèle organisationnel devraient être lui-même capable d'évoluer en cours de fonctionnement. Pour ce faire, les propriétés du systèmes (section 2.3) pourraient être utilisées comme critères pour l'évaluation de l'organisation et ainsi la gestion de l'organisation pourrait se mettre en œuvre comme un problème d'optimisation de contraintes distribuées [20].

Que changer dans l'organisation ? Dans le modèle de gouvernance présenté, l'adaptation de l'organisation est possible au niveau des spécifications structurelles, fonctionnelles et normatives. La réorganisation structurelle peut consister à ré-affecter des rôles aux agents, redéfinir des rôles et des cardinalités voire les interactions entre rôles. Quand de nouvelles applications sont déployées au sein du systèmes M2M, de nouveaux buts et schémas sociaux pourraient être définis comme des instances de buts et schémas génériques. Mais la spécification fonctionnelle elle-même pourrait également devoir être revue et les missions et plans devraient être adaptés en conséquence. Enfin, en modifiant les normes, le système pourrait changer l'affectation des missions aux rôles, renforcer ou relâcher certaines relations déontiques et adapter les missions aux nouvelles conditions et échéances.

Qui change l'organisation ? Afin de permettre au système d'être réorganisé, il est nécessaire de définir qui est responsable de tels changements. Si l'organisation est représentée par des artefacts spécifiques, les agents peuvent manipuler directement les spécifications. Ceci peut se faire grâce à des agents dédiés qui surveillent le fonctionnement du système et applique des politiques de réorganisation [7] ou en permettant aux agents applicatifs d'adapter directement le système suivant des indicateurs locaux, comme les situations non-coopératives des AMAS [1]. Cependant, un système large échelle semble difficile à surveiller par un système externe centralisé qui manque de robustesse alors que la deu-

xième approche augmente la charge des agents lors des prises de décisions organisationnelles et pose des problèmes de confiance. Combiner les deux approches pourrait allier les atouts complémentaires deux approches [13]. Entre autre, les agents applicatifs pourraient détecter des situations imprévues ou des spécifications inadaptées et proposer des adaptations à court terme alors que les agents de surveillance pourraient valider (ou non) ces changements organisationnels et ainsi réguler le degré de réorganisation du système en fonction des exigences originelles.

6 Conclusion et travaux futurs

Au travers d'une application de gestion intelligente de stationnement urbain, cet article a présenté une architecture multi-agent pour la gouvernance agile de systèmes M2M, suivant les dernières recommandations du groupe technique M2M de l'ETSI [4]. Afin de prendre en compte des exigences d'extensibilité, il souligne plusieurs points clés pour adapter les spécifications structurelles, fonctionnelles et normatives.

La prochaine étape de ce travail explorera les aspects de réorganisation suivant deux directions : (i) la spécification comportementale permettant aux agents de s'adapter à l'organisation et (ii) la définition de nouveaux rôles spécifiques à la surveillance de l'organisation et au processus de réorganisation. Entre temps, l'organisation, les agents et les artefacts proposés seront déployés dans une infrastructure M2M comme démonstrateur afin de tester et valider ce modèle en environnement réel.

Références

[1] C. Bernon, V. Camps, M.-P. Gleizes, and G. Picard. *Engineering Self-Adaptive Multi-Agent Systems : the ADELFE Methodology*, chapter 7, pages 172–202. Idea Group Publishing, 2005.

[2] R. Bordini, J. F. Hübner, and M. Wooldridge. *Agent Systems in AgentSpeak using Jason. Wiley Series in Agent Technology.* John Wiley & Sons, wiley seri edition, 2007.

[3] A.D. Chave, M. Arrott, C. Farcas, E. Farcas, I. Krueger, M. Meisinger, J.A. Orcutt, F.L. Vernon, C. Peach, O. Schofield, and J.E. Kleinert. Cyberinfrastructure for the US Ocean Observatories Initiative : Enabling interactive observation in the ocean. *Oceans 2009-Europe*, pages 1–10, May 2009.

[4] ETSI. ETSI TS 102 690 V<0.12.3>, Machine-to-Machine communications (M2M) ; Functional architecture. Technical report, ETSI, June 2011.

[5] B. Horling and V. Lesser. A survey of multi-agent organizational paradigms. *The Knowledge Engineering Review*, 19(04) :281–316, November 2005.

[6] J. F. Hübner, J. S. Sichman, and O. Boissier. A Model for the Structural, Functional, and Deontic Specification of Organizations in Multiagent Systems. In *Proc. of the 16th Brazilian Symposium on Artificial Intelligence (SBIA'02)*, number ii in LNCS, pages 118–128, Berlin, 2002. Springer.

[7] J. F. Hübner, J. S. Sichman, and O. Boissier. Using the MOISE+ for a cooperative framework of MAS reorganisation. In *Proc. of the 17th Brazilian Symposium on Artificial Intelligence (SBIA'04)*, pages 506–515, Berlin, 2004. Springer.

[8] N. R. Jennings. An agent-based approach for building complex software systems. *Communications of the ACM*, 44(4) :35–41, 2001.

[9] R. Kota, N. Gibbins, and N.R. Jennings. Decentralised Approaches for Self-Adaptation in Agent Organisations. *ACM Transactions on Autonomous and Adaptive Systems*, pages 1–36, 2011.

[10] A. Lefauconnier and E. Gantelet. La recherche d'une place de stationnement : strategies, nuisances associees, enjeux pour la gestion du stationnement en france.

[11] R Levy, W Chen, and M Lyell. Software agent-based framework supporting autonomous and collaborative sensor utilization (aamsrt). In *The 8th International Conference on Autonomous Agents and Multiagent Systems*, 2009.

[12] X. Ma and W. Luo. The Analysis of 6LowPAN Technology. In *2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application*, number 3 in PACIIA, pages 963–966. IEEE, December 2008.

[13] L. Penserini, V. Dignum, A. Staikopoulos, H. Aldewereld, and F. Dignum. Balancing Organizational Regulation and Agent Autonomy : An MDE-based Approach. In Huib Aldewereld, Virginia Dignum, and Gauthier Picard, editors, *Engineering Societies in the Agents World X*, pages 197–212. Springer Link, 2009.

[14] C. Persson, G. Picard, F. Ramparany, and O. Boissier. Problématique du passage à l'échelle dans les systèmes Machine-to-Machine. Technical report, Institut Fayol, 2011.

[15] G. Picard, J. F. Hübner, O. Boissier, and M.-P. Gleizes. Réorganisation et auto-organisation dans les systèmes multi-agents. In *Journées Francophones sur les Systèmes Multi-agents (JFSMA'09)*, pages 89–98. Cépaduès, 2009.

[16] G. Picard, S. Mellouli, and M.-P. Gleizes. Techniques for Multi-agent System Reorganization. In *Engineering Societies in the Agents World VI*, pages 142 – 152, Kusadasi, Turkey, 2006. Springer Link.

[17] M. Piunti, O. Boissier, J. F. Hübner, and A. Ricci. Embodied Organizations : a unifying perspective in programming Agents, Organizations and Environments. In *Coordination, Organizations, Institutions and Norms in Agent Systems*, pages 98–114. Springer, September 2010.

[18] A. Ricci, M. Piunti, and M. Viroli. Environment programming in multi-agent systems : an artifact-based perspective. *Autonomous Agents and Multi-Agent Systems*, pages 1–35, 2010.

[19] G. Serugendo, M.-P. Gleizes, and A. Karageorgos. Self-Organisation and Emergence in MAS : An Overview. *Informatica*, 30 :45–54, 2006.

[20] M. Yokoo. *Distributed Constraint Satisfaction : Foundations of Cooperation in Multi-Agent Systems*. Springer, 2001.