Contextualiser l'interaction entre agents en combinant dimensions sociale et physique au sein de l'environnement

S. Galland[a] stephane.galland@utbm.fr
N. Gaud[a] nicolas.gaud@utbm.fr
S. Rodriguez[b] sebastian.rodriguez@citat.org.ar
F. Balbo[c] flavien.balbo@mines-stetienne.fr
G. Picard[c] gauthier.picard@mines-stetienne.fr
O. Boissier[c] olivier.boissier@mines-stetienne.fr

[a]Laboratoire IRTES-SET, Institut IRTES, Université de Technologie de Belfort-Montbéliard, 90010 Belfort, France

[b]Laboratoire GITIA, Facultad Regional Tucumán, Universidad Tecnológica Nacional, San Miguel de Tucumán, CPA T4001JJD, Argentine

[c]Institut FAYOL, École Nationale Supérieure des Mines, 42000 Saint-Étienne, France

Résumé

L'environnement, en tant qu'espace partagé entre agents, est un élément essentiel des systèmes multiagents. Selon les systèmes, cet espace intègre des dimensions différentes comme une dimension physique support à l'ancrage spatial et à l'activité des agents sur cette dimension, ou une dimension sociale support aux communications entre agents. Ces dimensions sont souvent traitées de manière indépendante et ne sont reliées qu'au sein de l'agent qui constitue alors le lieu de jonction et de combinaison des informations véhiculées dans ces différentes dimensions. Il s'avère cependant que la combinaison entre ces dimensions est à considérer également en dehors des agents, pour pouvoir par exemple, situer des communications. Dans cet article, nous proposons un modèle unifié assurant la combinaison des dimensions physiques et sociales pour la mise en œuvre d'interactions contextualisées entre agents. Ce modèle est développé avec le langage multiagent SARL. Nous illustrons cette proposition par une application de simulation de trafic routier dans la ville de Belfort.

Mots-clés : *Environnement support à l'interaction, Environnement physique, Environnement Social, Langage de programmation, Trafic routier*

Abstract

The environment, as a space shared between agents, is a key component of multiagent systems. Depending on systems, this space integrates different dimensions like a /physical/ one, supporting space anchoring and agents' activities on these dimension, or a /social/ one, supporting agents' communications. These dimensions are generally considered independently and are only linked within the agent mind, which is therefore the place to join and combine information conveyed in these different dimensions. However, it appears that the combination of these dimensions is also considered outside the agents' mind, e.g. to locate communications. In this paper, we propose a unified model that supports the combination of physical and social dimensions to implement contextualized interactions between agents. This model is developed using the SARL multiagent language. We illustrate this approach by a traffic simulation, in the city of Belfort.

Keywords: *Environment as interaction support, Physic environment, Social environment, Programming language, Road traffic*

1 Introduction

L'environnement est à présent considéré comme une abstraction de premier ordre pour la conception de systèmes multiagents (SMA) [28]. Il a notamment été montré qu'il est possible de prendre avantage de l'environnement afin d'améliorer les possibilités d'interaction entre agents [19, 20, 25]. Il devient ainsi possible d'utiliser les propriétés d'observabilité de l'environnement afin d'assurer un partage des interactions. Cependant les termes *environnement* et *interaction* peuvent être déclinés différemment selon la nature du SMA. Ainsi, l'environnement peut être physique et l'interaction entre agents est alors indirecte par une modification de l'environnement. L'environnement peut également être social et l'interaction est un échange de messages entre agents. La nature multidimen-

Publié dans les actes des 22èmes Journées Francophones sur les systèmes multi-agents; Rémy Courdier et Jean-Paul Jamont (eds.); pp. 65–74; ISBN 978-2-36493-154-1.

sionnelle de l'environnement introduit des interprétations différentes du concept d'interaction. L'agent est alors le lieu de jonction et de combinaison des informations véhiculées dans les différentes dimensions où il interagit. Dans cette dernière vision, il n'est pas possible d'exploiter la combinaison de ces informations en dehors de l'agent pour enrichir son contexte interactionnel. Par exemple, l'émission d'un message se concrétise dans la dimension sociale en une interaction qui peut être contrainte par des conditions physiques (l'agent récepteur a-t-il la capacité physique de recevoir le message ?), ou une interaction dans la dimension physique peut être contrainte par des conditions sociales (un agent est suffisamment proche pour entendre le message mais n'a pas le droit d'y accéder). Il est également difficile de considérer une interaction selon ses différentes interprétations. Par exemple, un message émis dans la dimension sociale peut avoir des conséquences en fonction de son contenu dans sa dimension d'origine, et de sa forme dans la dimension physique (bande de fréquence dédiée aux urgences, par exemple). Dans cet article, nous proposons un modèle permettant la combinaison des dimensions physiques et sociales pour la mise en œuvre d'interactions contextualisées entre agents. Ce modèle est développé avec le langage multiagent SARL. Ce langage s'appuie directement sur les concepts multiagents et donne ainsi l'opportunité de prendre en compte les différents modèles d'environnement au sein d'un même langage.

La suite de l'article est organisée de la manière suivante. En section 2, nous présentons les différents services associés à l'environnement et comment l'interaction est prise en compte. Dans la section 3, notre exemple illustratif de gestion du trafic est introduit. En section 4, nous présentons comment le langage SARL peut modéliser l'environnement dans ses dimensions physiques et sociales. La modélisation de l'environnement combiné, et son application dans notre exemple de simulation de trafic, est détaillée dans la section 4.4. En section 6, nous discutons notre proposition vis-à-vis des autres travaux du domaine. Nous concluons enfin et donnons les perspectives envisagées pour ce travail en section 7.

2 Qu'est-ce que l'environnement ?

De nombreux travaux, notamment dans le groupe de travail E4MAS [1] (« Environment for multi-agent systems »), ont concerné la définition et le rôle de l'environnement dans la conception d'un SMA. Nous nous appuierons sur la définition de l'environnement issue de ces travaux : *The environment is a first-class abstraction that provides the surrounding conditions for agents to exist and that mediates both the interaction among agents and the access to resources.* Dans cette définition, l'environnement est à la fois un support d'exécution et à l'interaction. Il peut alors être considéré comme l'une des parties essentielles d'un système multiagent. Plus précisément les responsabilités suivantes peuvent être assurées par l'environnement [24, 28] :

Structuration
L'environnement permet de structurer le SMA aussi bien physiquement que socialement, et prend en charge les communications.

Gestion du cycle de vie
L'environnement est en charge de la maintenance des ressources et des agents, ainsi que de leur dynamique d'exécution.

Observabilité
L'environnement fournit les structures de représentation, ressources et "corps" des agents.

Accessibilité
L'environnement gère l'accessibilité aux structures et aux ressources.

Régulation
L'environnement contrôle les accès aux structures et aux ressources, et maintient les lois d'accès.

Ontologie
L'environnement fournit une ontologie commune qui peut être consultée par les agents

Si la définition de l'environnement ne pose pas de restriction, nous constatons que de nombreuses mises en œuvre existent selon une dimension choisie pour l'environnement : l'environnement physique [1, 8, 15, 28], l'environnement social [25], et l'environnement d'exécution [22]. Nous considérons les deux premiers environnements dans cet article. La notion d'« environnement physique » se réfère à la classe de systèmes réels ou simulés dans laquelle les agents, ainsi que les objets, disposent d'une position explicite et produisent des actions elles aussi localisées [13]. L'environnement social supporte les interactions selon différentes modalités directes, indirectes ou d'écoute

1. https://distrinet.cs.kuleuven.be/events/e4mas/

Publié dans les actes des 22èmes Journées Francophones sur les systèmes multi-agents; Rémy Courdier et Jean-Paul Jamont (eds.); pp. 65–74; ISBN 978-2-36493-154-1.

flottante entre les agents en proposant des mécanismes de diffusion et d'accès à des informations. L'environnement fournit alors l'observabilité, l'accessibilité et la structuration du SMA selon des règles sociales qui restreignent les interactions. Cette similitude fonctionnelle se traduit cependant par des modélisations différentes qui rendent difficile une utilisation transversale des fonctionnalités de l'environnement permettant d'utiliser conjointement les informations et événements issus des différentes dimensions de l'environnement.

Nous nous intéressons donc, dans cet article, à l'articulation entre les dimensions physiques et sociales, suivant trois cas d'utilisation conjoints :

Cas a : une **interaction dont la perception dans une dimension est contrainte par la seconde**. Par exemple, la réception du message dans l'environnement social est contrainte par des règles physiques (proximité).

Cas b : un **même événement génère des interactions différentes dans les deux dimensions**. Par exemple, le contenu du message modifie le comportement des membres d'une communauté et la forme du message modifie leur environnement physique.

Cas c : une **interaction dans une dimension qui génère d'autres interactions dans la seconde**. Par exemple, le message est issu de l'environnement social et modifie l'environnement physique.

3 Exemple illustratif

Pour illustrer notre proposition, nous simulons la régulation de traffic selon un modèle coopératif basé sur des outils embarqués de communication inter-véhicules ("*Vehicular Ad-Hoc Network*" ou Vanet). Nous faisons l'hypothèse que l'environnement physique est constitué d'un ensemble d'arcs équipés de capteurs pour connaître la densité du trafic, et d'intersections où des feux sont équipés pour relayer des messages. L'environnement social comprend les véhicules membres d'une communauté et équipés pour recevoir des messages. Similairement à [3], la régulation met en œuvre un plan dynamique de feux qui détermine les axes qui doivent être favorisés (durée du vert) selon l'importance du trafic (information issue de l'environnement physique) et la présence de véhicules prioritaires (information issue de l'environnement social). Dans cet environnement, nous traitons le cas d'un véhicule prioritaire qui veut la priorité et qui envoie régulièrement un message à la communauté d'utilisateurs pour demander qu'on lui cède le passage. Ce message contient un indice de priorité qui indique l'importance accordée par le véhicule à sa demande. À proximité d'un carrefour, ce message est reçu et relayé par le feu de signalisation. Ce scénario, nous permet d'illustrer les trois cas précédents d'utilisation conjoints :

Cas a : Un feu ne prend en compte que les demandes issues des véhicules prioritaires (statut social de l'émetteur).

Cas b : Le contenu du message modifie le comportement des membres de la communauté et la forme du message permet une augmentation de l'indice de l'axe routier concerné et, par conséquent, une interaction indirecte avec les autres véhicules.

Cas c : Une collision dans l'environnement physique provoque l'envoi d'un message d'alerte dans l'environnement social. Il sera pris en compte par les véhicules de secours pour planifier leur intervention.

4 Modélisation des deux dimensions de l'environnement avec SARL

Nous allons présenter dans cette section comment décrire les dimensions physiques et sociales de l'environnement, dans le langage de développement multiagent SARL.

4.1 SARL : langage orienté-agent

SARL [2] est un langage orienté-agent généraliste [22]. Il vise à fournir les abstractions fondamentales pour supporter la concurrence, la distribution, l'interaction, la décentralisation, la réactivité, l'autonomie et la reconfiguration dynamique d'agents. Ces fonctionnalités de haut niveau sont désormais considérées comme les principales exigences pour une mise en œuvre simple et pratique d'applications logicielles orientées-agent. La principale perspective qui a guidé la création de SARL est la création d'un langage ouvert et facilement extensible. Un tel langage doit fournir un ensemble réduit de concepts essentiels à la mise en œuvre d'un

2. `http://www.sarl.io`

Publié dans les actes des 22èmes Journées Francophones sur les systèmes multi-agents; Rémy Courdier et Jean-Paul Jamont (eds.); pp. 65–74; ISBN 978-2-36493-154-1.

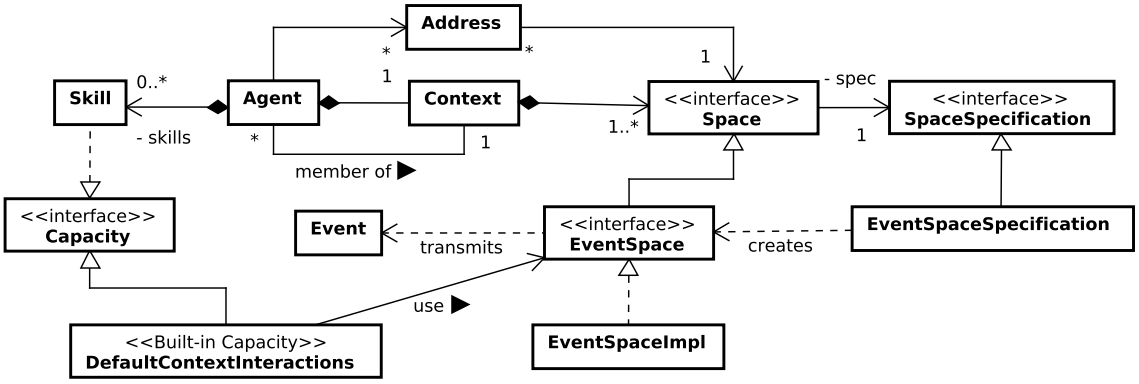

FIGURE 1 – Concepts principaux de SARL

système multiagent. Le métamodèle de SARL est basé sur quatre concepts principaux : Agent, Capacité, Espace et Adresse (illustrés par la figure 1).

Un agent (`Agent`) est une **entité autonome exhibant un comportement**. Ce comportement peut nécessiter l'**utilisation de capacités** (`Capacity`) de l'agent. La réalisation concrète d'une capacité est assurée par une compétence (`Skill`) possédée par l'agent. Ces deux derniers concepts ont été initialement proposés dans le métamodèle organisationnel CRIO [5]. Un agent doit pouvoir interagir avec d'autres agents ou avec son environnement. Il existe de nombreuses modélisations de l'interaction [16] : messages, événements, influence-réaction, etc. Pour permettre la modélisation de ces interactions, le langage SARL propose le concept d'espace (`Space`) décrivant un **espace d'interaction**, dans lequel chaque agent possède une adresse. Dans le langage SARL, la définition d'un espace concret est proposée par défaut. Il s'agit d'un espace permettant de propager des événements, nommé `EventSpace` (et son implantation `EventSpaceImpl`). De nouveaux espaces d'interaction peuvent être définis en créant une spécification (`SpaceSpecification`) décrivant les attributs, les fonctions et les propriétés de ces espaces.

SARL est intégré dans l'environnement de développement Eclipse, et la nouvelle version de la plate-forme JANUS [3] supporte pleinement les concepts mentionnés ci-dessus.

L'espace, au sens SARL, est l'abstraction permettant de définir un **espace d'interaction entre des agents, ou entre des agents et un environnement**. Nous utilisons ce concept pour définir les interactions entre un agent et l'environnement dans le cadre des dimensions physiques et sociales de ce dernier.

4.2 Dimension physique de l'environnement

La dimension physique de l'environnement contient un ensemble d'objets, incluant les corps des agents [8]. Par abus de langage, nous qualifierons cette dimension d'« environnement physique » dans la suite de cet article. L'environnement physique est considéré comme inaccessible, non déterministe, dynamique et continu. Il peut être hiérarchiquement décomposé en un ensemble de zones, sous-zones et ainsi de suite [6], elles-mêmes connectées entre elles par un lien de voisinage. Cela donne naissance à une structure complexe qui peut être considérée comme une sorte de "*clustered graph*" ou hypergraphe. Chaque zone est associée à l'ensemble des objets y étant localisés. L'environnement peut également être structuré sous la forme d'un graphe représentant un réseau routier, lui-même constitué de segments de route (`Segment`) interconnectés. Cette approche est utilisée dans le cadre de notre exemple illustratif, dans la section 4.4. Nous ne détaillerons pas ce modèle d'environnement. Le lecteur intéressé peut se référer à [7, 8].

Afin de permettre à l'agent d'interagir avec l'environnement physique, il est nécessaire de définir une capacité dédiée. Le script 1 décrit les fonctionnalités accessibles à un agent pour tous les environnements physiques (`AbstractPhysicEnvironmentCapacity`), et plus particulièrement pour les environ-

3. `http://www.janusproject.io`

Publié dans les actes des 22èmes Journées Francophones sur les systèmes multi-agents; Rémy Courdier et Jean-Paul Jamont (eds.); pp. 65–74; ISBN 978-2-36493-154-1.

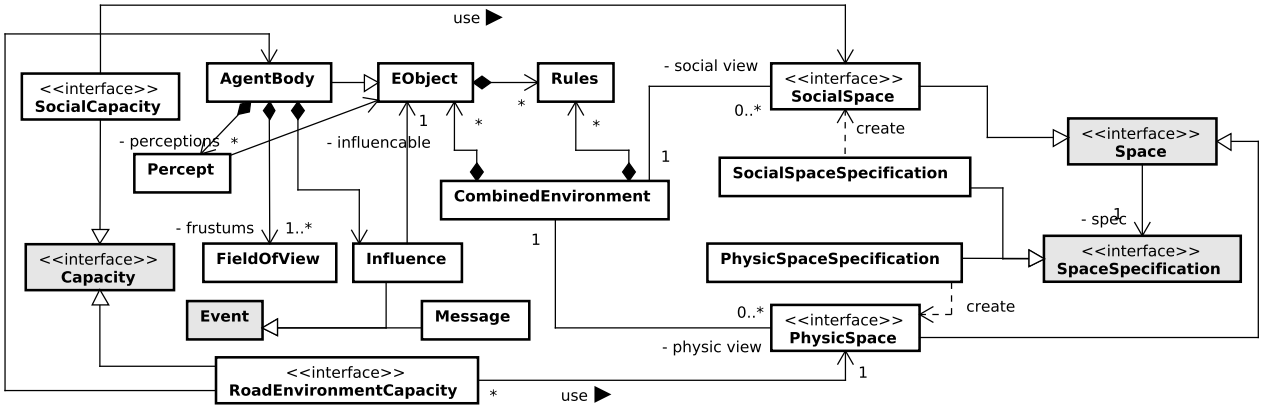

FIGURE 2 – Concepts relatifs au modèle d'environnement (en blanc)

nements constitués de réseaux routiers (`RoadEnvironmentCapacity`). Nous proposons cette distinction afin de faciliter la définition d'autres capacités dédiées à des environnements 2D et 3D (qui utilisent des objets mathématiques différents).

```
capacity AbstractPhysicEnvironmentCapacity {
  def getLinearSpeed : double
  def setPhysicalPerceptionAlterator(filter :
      PhysicalPerceptionAlterator)
  def setInterestFilter(filter : InterestFilter)
  def influence(inf : Influence)
  def killBody
}
capacity RoadEnvironmentCapacity extends
    AbstractPhysicEnvironmentCapacity {
  def getPosition : Pair<Segment,double>
  def getOrientation : Direction
}
```

Script 1 – Capacités d'interaction avec l'environnement physique (en SARL)

Un agent peut agir dans l'environnement physique, via son corps (`AgentBody`), en utilisant le modèle d'interaction Influence-Réaction [13]. Une influence peut décrire un mouvement (`MotionInfluence`) ou l'exécution d'une action sur un objet (`ActionInfluence`). Nous considérerons que **les influences sont des événements ayant un impact sur un objet de l'environnement** (voir figure 2). Ces deux types d'influences sont décrits par des événements reçus par l'environnement (script 2).

```
event Influence {
  var object : EObject
}
event MotionInfluence extends Influence {
  var linearSpline : double
  var linearShift : double
  var path : Segment[]
}
event ActionInfluence extends Influence {
  var actionName : String
  var actionParameters : Object[]
}
event Perception {
  var objects : Percept[]
}
```

Script 2 – Événements liés à l'environnement physique (en SARL)

Le mécanisme de perception d'un agent dans l'environnement physique est soutenu par le **corps de l'agent** (`AgentBody`) qui contient entre autres une description géométrique du champ de perception (`FieldOfView`). L'agent peut affiner le calcul de ses perceptions en fournissant une description permettant d'altérer les propriétés physiques de ses capteurs (`PhysicalPerceptionAlterator`), et/ou un filtre décrivant les objets qui l'intéressent (`InterestFilter`). Le premier ne tiendra compte que des altérations liées à la physique des capteurs, le second que de la sémantique des objets perçus. Une fois que l'ensemble des objets se trouvant dans le champ de perception de l'agent est déterminé, il est fourni à ce dernier par un événement de type `Perception`.

Pour chaque capacité décrite, une compétence concrète doit être définie. Cette compétence permettra de créer l'espace d'interaction (script 3), et d'invoquer les fonctions associées à cet espace. Un espace dédié à l'environnement physique doit permettre de créer un corps d'agent, de déposer ce corps dans l'environnement, et de notifier l'environnement de l'existence d'influences émises par un corps.

```
public interface PhysicSpace extends Space {
  public PhysicBodyFactory getBodyFactory();
  public void putInEnvironment(AgentBody body,
      Agent perceptionListener);
  public void influence(AgentBody body, influences
      : Influence[]);
  public void destroyBody(AgentBody body);
}
public class PhysicSpaceImpl extends
    EventSpaceImpl implements PhysicSpace {
```

Publié dans les actes des 22èmes Journées Francophones sur les systèmes multi-agents; Rémy Courdier et Jean-Paul Jamont (eds.); pp. 65–74; ISBN 978-2-36493-154-1.

```
  private CombinedEnvironment env;
  public void influence(AgentBody body, influences
       : Influence[]) {
    for(Influence i : influences) emit(i, new
         Scope(env));
  }
  ...
}
```

Script 3 – Spécification de l'espace physique (en Java)

L'implantation de l'espace lié à l'environnement physique (présenté dans le script 3) est dépendante du modèle concret choisi de l'environnement, par exemple JaSim [7, 8]. Nous proposons que chaque influence émise par un agent soit envoyée uniquement à l'entité environnement, qui la traitera dans son comportement propre. Pour cela, nous utilisons le mécanisme d'envoi de messages et d'événements proposé par le langage SARL (un message est considéré comme un événement du point de vue de son récepteur). Nous restreignons l'ensemble des récepteurs de l'événement par la création d'une portée (`Scope`).

L'environnement physique considéré dans cet article est un environnement simulé. Ce choix a été fait afin de faciliter les expérimentations. Toutefois, il est possible de connecter les agents au monde réel en leur fournissant un outil logiciel représentant leurs corps dans cet environnement physique. Ce corps peut alors proposer des capteurs et des effecteurs dans un univers virtuel ou dans le monde réel. Par exemple, l'agent capable de conduire un véhicule autonome proposé par [10] a été utilisé à la fois dans un simulateur et dans un véhicule réel : seules les implantations des capteurs et des effecteurs associés au corps de l'agent (`AgentBody` dans la figure 2) ont été modifiées.

4.3 Dimension sociale de l'environnement

SARL permet de définir un type particulier d'espace appelé `SocialSpace` qui fournit un support aux interactions sociales entre les agents. Dans ce type d'espace, les agents communiquent en échangeant des messages à l'aide de la capacité décrite dans le script 4.

```
event Message {
  var destination : Scope<Address>
}
capacity SocialCapacity {
  def emit(e : Message, scope : Scope<Address> =
       null)
}
```

Script 4 – Capacités d'interaction sociale (en SARL)

Considérons le modèle d'interaction proposé par défaut par le langage SARL (interface `EventSpace` et son implantation `EventSpaceImpl`). Il utilise les événements comme support de l'interaction : **un message devient un événement pour le ou les agents devant le recevoir.** Dès lors, l'environnement social peut être considéré comme une spécialisation d'un espace d'échange d'événements, et assurant le lien vers l'entité environnement.

```
public interface SocialSpace extends Space {
  public void emit(Message e, Scope<?> scope);
  public Address register(Agent agent);
  public void unregister(Address agentAddress);
}
public class SocialSpaceImpl extends
     EventSpaceImpl implements SocialSpace {
  private CombinedEnvironment env;
  public void emit(Message e, Scope<?> scope) {
    e.destination = scope;
    super.emit(e, new Scope(env));
  }
}
```

Script 5 – Spécification de l'espace social (en Java)

Le script 5 fournit la définition d'un espace social (`SocialSpace`) et d'une implantation possible (`SocialSpaceImpl`) basée sur les outils de SARL. La section 4.4 précise le traitement de ces événements par l'environnement.

4.4 Environnement combiné

Les deux vues sur l'environnement présentées dans la section précédente sont supportées et combinées au sein d'une même entité (`CombinedEnvironment`). La figure 2 illustre les relations entre cette entité et les vues associées aux deux dimensions considérées dans cet article.

Le comportement général de l'environnement est une combinaison des comportements associés à ses deux dimensions : (i) calculer les réactions correspondant à la modification de l'état de l'environnement physique à partir d'influences émises par les agents ou par l'environnement lui-même ; (ii) calculer les perceptions dans la dimension physique ; (iii) propager ou router les messages dans la dimension sociale. Ces différents points ont été modélisés par quatre fonctions invoquées à chaque pas de temps dans le simulateur (événement `SimulationStep` dans le script 6). Les modèles implantés dans ces fonctions sont décrits dans [2, 8].

```
behavior CombinedEnvironment {
  var rules : List<Pair<(Behavior, Event, Object)
       => boolean, (Behavior, Event, Object) =>
       boolean>>
```

Publié dans les actes des 22èmes Journées Francophones sur les systèmes multi-agents; Rémy Courdier et Jean-Paul Jamont (eds.); pp. 65–74; ISBN 978-2-36493-154-1.

```
3     on Initialize {
4       every(500) [ wake(new SimulationStep) ]
5     }
6     on SimulationStep {
7       computeEndogenousInfluences
8       computePhysicReactionsFromInfluences
9       computePhysicPerceptions
10      deliverMessages
11    }
12    on Influence {
13      if (applyRules(occurrence, occurrence.object))
              {
14        saveInfluence(occurrence)
15      }
16    }
17    on Message {
18      for(participant : this.socialSpace.
              participants) {
19        if (occurrence.scope.matches(participant)) {
20          if (applyRules(occurrence, participant)) {
21            saveMessage(occurrence)
22          }
23        }
24      }
25    }
26    def applyRules(e : Event, o : Object) : boolean
          {
27      for(pair : rules) {
28        if (pair.first(this, e, o)
29            && !pair.second(this, e, o)) {
30          return false
31        }
32      }
33      return true
34    }
35  }
```

Script 6 – Comportement de l'environnement (en SARL)

Afin de permettre les trois cas d'utilisation décrits dans la section 2, **un ensemble de règles est stocké dans l'environnement**. Chaque règle est constituée de : (i) un prédicat p permettant de décrire la condition d'activation de la règles ; et (ii) une fonction f décrivant les actions à réaliser lorsque la règle est activée. **L'expression concrète du prédicat peut dépendre de l'état de l'environnement dans une ou plusieurs de ses dimensions, et/ou des états d'objets particuliers** (objets physiques ou messages). Nous proposons d'utiliser une clôture syntaxique pour permettre la définition de p et de f. Une clôture est une fonction qui capture des références à des variables libres dans l'environnement lexical de la fonction dans laquelle elle est définie. Cette fonctionnalité permet de définir simplement des fonctions à l'intérieur d'autres fonctions. Ces clôtures peuvent alors être passées en paramètres de fonctions, par exemple. Dans SARL, la notation pour déclarer une clôture est `(P) => R`, où `P` est la liste des types des paramètres formels, et `R` est le type des valeurs retournées. La création de ces règles doit prendre en compte la priorisation d'une règle par rapport à une autre. Nous proposons de les placer dans **une liste de la plus prioritaire à la moins prioritaire**. Dans le script 6, nous définissons l'ensemble des règles (variable *rules*) comme une liste de paires (p, f). p est un prédicat prenant la forme d'une clôture avec comme paramètres un `Behavior` (abstraction pour tout algorithme exhibant un comportement dynamique, y compris les agents), un événement dans une des dimensions de l'environnement, un objet concerné par l'événement, et retournant une valeur booléenne indiquant si le prédicat (la règle) est activable ou non. f est une clôture prenant les mêmes paramètres que p, et réalisant la tâche associée à la règle et retournant une valeur booléenne indiquant si la propagation de l'influence ou du message dans sa dimension d'origine est autorisée.

Cet ensemble de règles est utilisé lorsqu'une influence est émise dans la dimension physique, ou un message dans la dimension sociale. Dans le script 6, ces deux cas sont distingués au sein de deux fonctions (`on Influence` et `on Message`) afin de fournir aux prédicats l'objet sur lequel ils pourront s'appliquer : l'objet physique « influencé » ou le message. Ces deux fonctions appliquent les règles connues par l'environnement (`applyRules`). Si aucune règle n'annule l'interaction, alors l'environnement la mémorise pour une utilisation ultérieure dans son comportement global (`SimulationStep`).

5 Extension du modèle pour la simulation de traffic

Le script 7 propose la définition des règles associées à notre exemple illustratif (section 3), ainsi que l'initialisation du système. La notation SARL `[ paramètres | instructions ]` permet de définir les clôtures associées à chaque règle.

Dans la dimension physique, l'environnement émet des occurrences de l'événement `PhysicCollision` lorsqu'il détecte une collision entre des véhicules. Cet événement contient la position de la collision et l'un des objets y participant. L'événement `PriorityRequest` représente le message envoyé par un agent à un feu tricolore pour lui demander la priorité de passage. Dans la dimension sociale, nous définissons l'événement `Alert` qui représente un message d'alerte dans la communauté.

```
1   agent Boot {
2     on Initialize {
3       var environment = new CombinedEnvironment(
              occurrence.parameters.get(0) as
              RoadNetwork)
4       /* Règle pour le cas a */
5       environment.rules.add(new Pair(
6         [ env,e,o | e instanceof PriorityRequest &&
                o instanceof TraficLight ],
```

Publié dans les actes des 22èmes Journées Francophones sur les systèmes multi-agents; Rémy Courdier et Jean-Paul Jamont (eds.); pp. 65–74; ISBN 978-2-36493-154-1.

```
7         [ env,e,o |
8           env.socialSpace.participant(e.source.ID)
                instanceof PriorityVehicle ]))
9       /* Règle pour le cas b */
10      environment.rules.add(new Pair(
11        [ env,e,o | e instanceof PriorityRequest ],
12        [ env,e,o |
13          var sp = env.physicSpace
14          for(road : sp.getRoadsBetween(sp.
                getEObject(o.ID).position, sp.
                getEObject(e.source.ID))) {
15            road.priorityIndex = road.priorityIndex
                  + e.priority
16          }
17          return true ]))
18      /* Règle pour le cas c */
19      environment.rules.add(new Pair(
20        [ env,e,o | e instanceof PhysicCollision ],
21        [ env,e,o |
22          env.emit(new Alert(e.position))
23          return false ]))
24      /* Create spaces */
25      var sp = currentContext.createSpace(
              SocialSpaceSpecification, UUID.random(),
              environment)
26      var pp = currentContext.createSpace(
              PhysicSpaceSpecification, UUID.random(),
              environment)
27      /* Spawning of the vehicles */
28      spawnVehicles(sp.ID, pp.ID)
29      killMe
30    }
31  }
```

Script 7 – Initialisation de la simulation (en SARL)

Le comportement des conducteurs est basé sur une adaptation des algorithmes "*Intelligent Driver Model*" pour déterminer l'accélération du véhicule, et D* pour planifier dynamiquement le chemin à suivre. Ces adaptations sont détaillées dans [7, 9]. Le résultat de la simulation à permis de réaliser la **preuve de concept** de notre modèle combinant deux dimensions de l'environnement. La figure 3 fournit une copie d'écran du simulateur. Un accident est représenté par le cercle rouge. L'existence de cet accident a été propagée dans la dimension sociale par un événement `Alert`. Les agents participant à la communauté et ayant reçu ce message sont reliés au lieu de la collision par un trait. Les feux tricolores sont représentés par des cercles au bord des voies routières. Les traits entre les véhicules et l'ambulance (cercle vert) indiquent que les véhicules perçoivent, visuellement ou auditivement, l'ambulance dans la dimension physique et décident de lui céder le passage à cause de sa position dans la dimension sociale. Cet événement permet à chaque agent de replanifier son trajet en condidérant la position de l'accident et les trajets des véhicules de secours propagés dans la dimension sociale. Ainsi, chaque conducteur (ou son GPS) a la possibilité de réagir plus tôt que ce que lui permet ses perceptions dans la dimension physique, et de participer à la réduction des congestions dans le système.

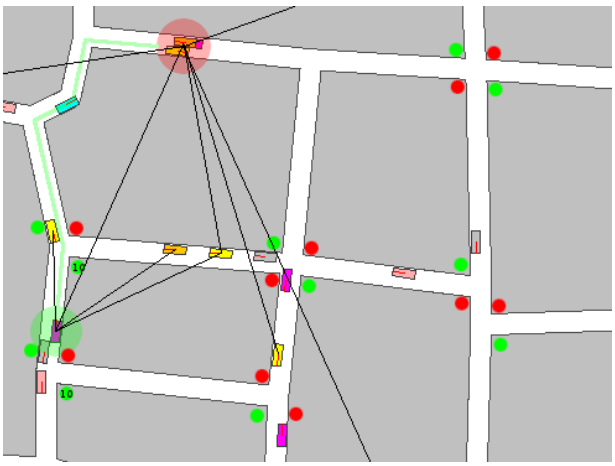

Figure 3 – Visualisation des dimensions physiques et sociales

6 Discussion

Selon nos connaissances, le modèle proposé dans cet article est la première tentative pour lier les différentes dimensions de l'environnement sans utiliser l'agent comme intermédiaire. Toutefois, de nombreux travaux ont déjà été menés sur la modélisation de l'environnement dans ses différentes dimensions.

Notre source d'inspiration pour l'environnement physique est le modèle d'environnement virtuel proposé par [8]. Il s'agit d'une adaptation et d'une spécialisation des modèles de [6, 28] pour la simulation de foules et de trafic en tenant compte des spécificités des environnements 3D virtuels. Le modèle CArtAgO [21] a également été une source d'inspiration en proposant le modèle d'interaction entre les agents et les artéfacts [20] dans l'environnement, et la définition de ces derniers.

La problématique de l'interaction entre un agent et l'environnement physique a été traitée selon plusieurs perspectives. L'un des modèles retenus dans notre approche est influence-réaction [13]. Il permet de prendre en compte la simultanéité des actions dans un environnement en considérant les interactions initiées par les agents comme incertaines, en détectant puis résolvant tout conflit entre deux interactions. Cette approche peut être rapprochée du concept d'artefact [20] qui propose de modéliser les objets appartenant à l'environnement. Ces derniers fournissent un ensemble d'actions pouvant être appliquées sur chacun d'entre eux. [26] a proposé un modèle similaire, appelé "*smart object*", appliqué aux en-

Publié dans les actes des 22èmes Journées Francophones sur les systèmes multi-agents; Rémy Courdier et Jean-Paul Jamont (eds.); pp. 65–74; ISBN 978-2-36493-154-1.

vironnements physiques virtuels. Cette vision est supportée par notre modèle grâce aux événements de type `ActionInfluence`. Nous distinguons les influences liées aux déplacements spatiaux (`MotionInfluence`) de celles dédiées au déclenchement d'actions (`ActionInfluence`) afin de permettre une spécification détaillée des paramètres de chacune d'entre elles. Le modèle IODA et son extension PADAWAN [17] permettent la modélisation des interactions avec les différentes dimensions de l'environnement en considérant que toutes les entités sont des agents. Notre modèle est partiellement compatible avec cet vision dans le cadre de la modélisation de la dimension physique : les corps des agents ne sont pas des agents. Nous proposons dans [8] d'agentifier l'entité environnement et pas ses composants physiques.

Concernant l'environnement social, notre source d'inspiration est le modèle d'environnement proposé par [1, 25, 29]. Dans ces modèles, l'environnement est un espace partagé dans lequel les agents déposent ou retirent les filtres décrivant le contexte de leurs interactions afin notamment de gérer les communications multiparties. Ces filtres sont alors gérés par l'environnement. L'environnement physique n'est pas dissocié de l'environnement social et les filtres concernent toujours un agent. Par conséquent, il est possible de traiter le cas d'utilisation a) présenté dans la section 2, mais pas explicitement les deux autres cas.

Certaines approches organisationnelles considèrent l'environnement. Par exemple, dans [11, 18], les auteurs proposent d'intégrer l'environnement dans un modèle organisationnel. L'élément dans la thématique de notre article concerne l'introduction du concept d'espace qui est une abstraction commune pour les groupes organisationnels et les zones spatiales. Toutefois, ces deux modèles ne proposent pas explicitement de considérer les dimensions physiques et sociales conjointement, ainsi que leurs interactions directes.

7 Conclusion et Perspectives

Dans cet article, nous proposons un modèle permettant la combinaison des dimensions physiques et sociales de l'environnement afin d'enrichir les capacités de modélisation de l'environnement et de fournir aux agents des outils de définition de comportements plus complexes. Nous proposons de définir les deux dimensions considérées à l'aide d'espaces d'interaction dans le langage SARL. Ces espaces peuvent être considérés comme des vues sur un environnement combinant les différentes dimensions. Afin de modéliser les relations entre ces dimensions, nous proposons de définir des règles. Elles permettent de modifier l'état de l'environnement lié à une dimension en fonction des éléments liés à l'autre dimension. Nous pensons que l'usage de règles fournit un outil assez général et adaptable pour différentes classes d'applications. Nous présentons les définitions concrètes de quelques règles dédiées à la simulation de trafic routier.

Une première perspective de nos travaux concerne la mise en correspondance ou l'adaptation de notre modèle pour permettre d'autres approches de modélisation de l'environnement : artefacts [20], "*smart objects*" [26], holarchies [8], etc.

L'enrichissement des capacités comportementales des agents peut être réalisé par un modèle d'environnement informé. Un environnement informé fournit une information sémantique détaillée aux agents [26] à l'aide de marqueurs sémantiques ou d'ontologies. Les premiers travaux proposant un modèle d'univers virtuel [6] ont été étendus dans le cadre de la simulation multiagent [2]. L'utilisation d'ontologies (et des outils associés) peut être considérée pour faciliter l'expression des règles d'interaction entre les différentes dimensions de l'environnement.

La nature intrinsèquement hiérarchique de l'environnement n'est pas traitée dans cet article. Certains auteurs ont déjà considéré la décomposition structurelle de l'environnement physique [6, 27], et sa relation avec son comportement dynamique, en utilisant une vision holonique de l'environnement [8]. Cette nature hiérarchique a été également modélisée dans la dimension sociale par des holarchies [23], des groupes agentifiés [14] ou des agents récursifs [4]. Notre modèle (`CombinedEnvironment`) peut être adapté pour correspondre à un holon (agent) décomposé en sous-holons.

Enfin, nous voulons faire évoluer les éléments syntaxiques du langage SARL pour faciliter la définition et la description des environnements et de leurs règles, à l'instar de travaux comme GAMA [12].

Références

[1] F. Badeig and F. Balbo. Définition d'un cadre de conception et d'exécution pour la simulation multi-

Publié dans les actes des 22èmes Journées Francophones sur les systèmes multi-agents; Rémy Courdier et Jean-Paul Jamont (eds.); pp. 65–74; ISBN 978-2-36493-154-1.

agent. *Revue d'Intelligence Artificielle*, 26(3) :255–280, 2012.

[2] F. Béhé, S. Galland, N. Gaud, C. Nicolle, and A. Koukam. An ontology-based metamodel for multiagent-based simulations. *International Journal on Simulation Modelling, Practice, and Theory*, 40 :64–85, January 2014.

[3] N. Bhouri, F. Balbo, and S. Pinson. An agent-based computational approach for urban traffic regulation. *Progress in AI*, 1(2) :139–147, 2012.

[4] K. C. Correa e Silva Fernandes. *Systèmes Multi-Agents Hybrides : Une Approche pour la Conception de Systèmes Complexes*. PhD thesis, Université Joseph Fourier- Grenoble 1, 2001.

[5] M. Cossentino, N. Gaud, V. Hilaire, S. Galland, and A. Koukam. ASPECS : an agent-oriented software process for engineering complex systems - how to design agent societies under a holonic perspective. *Autonomous Agents and Multi-Agent Systems*, 2(2) :260–304, March 2010.

[6] N. Farenc, R. Boulic, and D. Thalmann. An informed environment dedicated to the simulation of virtual humans in urban context. In *Proceedings of EUROGRAPHICS 99*, pages 309–318, 1999.

[7] S. Galland, J. Buisson, N. Gaud, M. Gonçalves, A. Koukam, F. Guiot, and L. Henry. Agent-based simulation of drivers with the JANUS platform. In *3nd International Workshop on Agent-based Mobility, Traffic and Transportation Models, Methodologies and Applications (ABMTRANS14)*. Springer, June 2014.

[8] S. Galland and N. Gaud. Holonic model of a virtual 3D indoor environment for crowd simulation. In *International Workshop on Environments for Multiagent Systems (E4MAS14)*. Springer, May 2014.

[9] S. Galland, A.-U.-H. Yasar, L. Knapen, N. Gaud, D. Janssens, O. Lamotte, G. Wets, and A. Koukam. Multi-agent simulation of individual mobility behavior in carpooling using the JANUS and JASIM platforms. *International Journal on Transport Research Part C*, 2014.

[10] F. Gechter, J.-M. Contet, O. Lamotte, S. Galland, and A. Koukam. Virtual intelligent vehicle urban simulator : Application to vehicle platoon evaluation. *Simulation Modelling Practice and Theory (SIMPAT)*, 24 :103–114, May 2012.

[11] A. Gouaïch and F. Michel. Towards a unified view of the environment(s) within multi-agent systems. *Informatica*, 29(4) :423–432, may 2005.

[12] A. Grignard, P. Taillandier, B. Gaudou, D. A. Vo, N. Q. Huynh, and A. Drogoul. GAMA 1.6 : Advancing the art of complex agent-based modeling and simulation. In *16th International Conference on Principles and Practices in Multi-Agent Systems (PRIMA)*, volume 8291, pages 242–258, Dunedin, New Zealand, 2013.

[13] F. Michel. The IRM4S model : the influence/-reaction principle for multiagent based simulation. In *Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS07)*. ACM, May 2007.

[14] J. Odell, M. Nodine, and R. Levy. A metamodel for agents, roles, and groups. In *Agent-Oriented Software Engineering (AOSE) IV*, Lecture Notes on Computer Science. Springer, 2005.

[15] J. Odell, H. Parunak, M. Fleisher, and S. Brueckner. Modeling Agents and their Environment. In *Agent-Oriented Software Engineering III*, volume 2585 of *Lecture Notes In Computer Science*, N.Y. (USA), 2002. Springer-Verlag.

[16] J. Peña, R. Levy, M. Hinchey, and A. Ruiz-Cortés. Dealing with complexity in agent-oriented software engineering : The importance of interactions. In *Conquering Complexity*, pages 191–214. Springer London, 2012.

[17] S. Picault, P. Mathieu, and Y. Kubera. PADAWAN, un modèle multi-échelles pour la simulation orientée interactions. In *JFSMA*, pages 193–202. Cépaduès, 2010.

[18] M. Piunti, A. Ricci, O. Boissier, and J. Hübner. Embodying organisations in multi-agent work environments. In *IEEE/WIC/ACM Int. Conf. on Web Intelligence and Intelligent Agent Technology (WI-IAT 2009)*, Milan, Italy., 2009.

[19] E. Platon, N. Sabouret, and S. Honiden. Tag interactions in multiagent systems : Environment support. In *Environment for Multi-Agent Systems (E4MAS)*, volume 4389 of *Lecture Notes in Artificial Intelligence*, pages 106–123. Springer Verlag, 2007.

[20] A. Ricci, M. Viroli, and A. Omicini. Programming MAS with artifacts. In *International Workshop on Programming Multi-Agent Systems*. Springer Verlag, July 2005.

[21] A. Ricci, M. Viroli, and A. Omicini. CArtAgO : A framework for prototyping artifact-based environments in MAS. In *E4MAS*. Springer Verlag, May 2007.

[22] S. Rodriguez, N. Gaud, and S. Galland. SARL : a general-purpose agent-oriented programming language. In *International Work on Intelligent Agent Technology (IAT)*, 2014. to be published.

[23] S. Rodriguez, V. Hilaire, N. Gaud, S. Galland, and A. Koukam. *Holonic Multi-Agent Systems*, chapter 11, pages 238–263. Self-Organising Software From Natural to Artificial Adaptation - Natural Computing. Springer, first edition, March 2011.

[24] J. Saunier. *Les communications multi-parties et leur régulation dans les systèmes multi-agents : modèle et support*. PhD thesis, Université Paris-Dauphine, 2006.

[25] J. Saunier, F. Balbo, and S. Pinson. A formal model of communication and context awareness in multiagent systems. *Journal of Logic, Language and Information*, pages 1–29, 2014.

[26] D. Thalmann and S. R. Musse. *Crowd simulation*. Springer, 2007.

[27] G. Thomas. *Environnements virtuels urbains : modélisation des informations nécessaires à la simulation de piétons*. PhD thesis, Informatique, Université de Rennes 1, 16 décembre 1999.

[28] D. Weyns, A. Omicini, and J. Odell. Environment as a first-class abstraction in multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 14(1) :5–30, February 2007. Special Issue on Environments for Multi-agent Systems.

[29] M. Zargayouna and F. Balbo. Langage de coordination multi-agent sécurisé. *Revue d'Intelligence Artificielle*, 27(3) :271–298, 2013.

Publié dans les actes des 22èmes Journées Francophones sur les systèmes multi-agents; Rémy Courdier et Jean-Paul Jamont (eds.); pp. 65–74; ISBN 978-2-36493-154-1.