Patterns multi-niveaux pour les SMA

A. Maudet[a] adrien.maudet@ign.fr
G. Touya[a] guillaume.touya@ign.fr
C. Duchêne[a] cecile.duchene@ign.fr
S. Picault[b] sebastien.picault@univ-lille1.fr

[a]Université Paris Est/IGN, COGIT, Saint-Mandé, France

[b]Équipe SMAC, CRIStAL, Université Lille 1, Villeneuve d'Ascq, France

Résumé

Depuis quelques années, les travaux sur les SMA multi-niveaux ont pris une importance croissante. Devant la diversité des modèles proposés, nous pensons qu'il est utile d'identifier des situations récurrentes et de les caractériser d'une manière suffisamment abstraite pour pouvoir comparer de manière formelle les modèles existants et faciliter la conception de nouveaux modèles. Dans ce but, nous proposons une première liste de patterns SMA multi-niveaux. Ces patterns sont issus d'un travail d'unification de modèles SMA multi-niveaux dédiés à la résolution d'un problème spatialisé (la généralisation cartographique). La structure et la dynamique de chaque pattern sont décrites formellement et accompagnées d'exemples issus d'une part du contexte de la généralisation cartographique, d'autre part d'autres contextes applicatifs multi-agents, en simulation notamment. Nous discutons également la possibilité de réutiliser et composer ces patterns.

Mots-clés : *SMA multi-niveaux, patterns, agents situés, conception de SMA, généralisation cartographique, simulation*

Abstract

Multi-level MAS have become a growing research topic in the last years. Due to the diversity of proposed models, we claim that recurrent situations have to be identified and characterized in an abstract way, in order to allow formal comparisons between existing models and to facilitate the design of new models. Therefore, we propose a first list of multi-level MAS patterns, coming from an attempt to unify several multi-level MAS models dedicated to a spatialized problem solving (cartographic generalization). The structure and dynamic of each pattern are formally described, and illustrated through examples drawn from the context of cartographic generalization and other MAS applications (especially in simulation). We also discuss the possibility to re-use and compose those patterns.

Keywords: *Multi-level MAS, patterns, situated agents, MAS design, cartographic generalisation, simulation*

1 Introduction

La recherche dans le cadre des systèmes multi-agents multi-niveaux connaît de récentes avancées. Ainsi, nombre de cas d'application impliquent la prise en compte de plusieurs niveaux, par exemple de simulation d'évolution de zones urbaines [7] ou encore des mouvements de foules [30]. Les aspects multi-niveaux communs à certains modèles ont été comparés dans [14]. Extraire de ces cas d'application concrets des situations génériques permet de mettre en commun des connaissances éprouvées, et d'enrichir la connaissance dans le domaine des systèmes multi-niveaux.

Plusieurs travaux ont mené à la formalisation des niveaux entre eux. Ces formalisations portent principalement sur l'aspect structurel des modèles (les liens entre les différents niveaux), ou sur des aspects spécifiques au passage d'un niveau à un autre. Dans [7], les phénomènes émergents sont réifiés en proposant l'introduction d'un élément d'interposition qui permet de contrôler les actions de l'agent créé à partir d'un phénomène émergent sur l'environnement de simulation. Dans [5], une formalisation est proposée pour les problèmes impliquant l'utilisation parallèle de deux modélisations à deux niveaux différents en proposant de définir des fonctions d'émergence et d'immergence permettant de maintenir la cohérence entre les deux niveaux. Dans [4], une formalisation de l'identification de groupes est proposée avec pour objectif de permettre une visualisation de ces derniers lors de différentes simulations. Cette formalisation est indépendante de la méthode de *clustering* choisie pour identifier un groupe. Dans [30], un méta-modèle est proposé pour exprimer la notion de capture et de libération d'agent par un agent de niveau supérieur. Lorsqu'il est capturé, un agent peut voir sa

position dans l'espace gérée par l'agent qui l'a capturé. Il récupère la gestion de son positionnement dans l'espace au moment de sa libération.

Toutes ces approches ont pour particularité de mettre l'accent sur le passage d'un niveau à un autre, que ce soit dans le cadre de la construction d'objet d'un niveau supérieur à partir d'agents d'un niveau inférieur, ou pour la transmission d'information du résultat d'interactions du niveau supérieur aux agents du niveau inférieur. Lors de l'étude de la modélisation de SMA pour un cas de résolution d'un problème spatialisé contraint, la généralisation cartographique, nous avons identifié que d'autres mécanismes plus précis se répétaient. L'objectif de cet article est de proposer une modélisation de ces situations sous forme de *patterns* génériques, en s'appuyant sur les cas concrets de la généralisation cartographique, et d'étudier dans quelle mesure ces *patterns* permettent de traiter des situations similaires rencontrées dans d'autres cadres applicatifs.

Le plan de cet article est le suivant : la partie 2 explicite nos motivations pour l'élaboration de *patterns*, et le cadre de la généralisation cartographique dans lequel ils ont été identifiés ; dans la partie 3, nous présentons le cadre théorique que nous utilisons dans la partie 4 pour la définition de ces patterns ; enfin, dans la partie 5 nous discutons nos propositions avant de conclure dans la partie 6.

2 Démarche et contexte applicatif

2.1 Motivation pour l'identification de patterns multi-niveaux

Lors de la résolution de problèmes multi-niveaux dans le domaine de la généralisation (présentée dans la section suivante), nous avons rencontré des motifs structurels ou comportementaux récurrents. Nous supposons que ces motifs sont susceptibles d'être utilisés pour l'analyse et la modélisation d'autres situations dans d'autres domaines, voire pour l'implémentation de solutions. Pour cela, nous essayons de décrire ces situations récurrentes d'une façon abstraite (en les caractérisant *a minima* et indépendamment du domaine d'où elles sont issues), d'identifier des exemples où elles apparaissent hors du domaine d'origine, et de montrer les liens qu'elles entretiennent les unes avec les autres.

À travers la notion de *pattern*, nous entendons

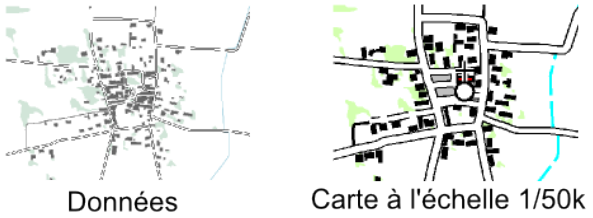


FIGURE 1 – Données initiales et carte généralisée.

construire une représentation abstraite minimale décrivant une situation rencontrée de façon récurrente, en nous affranchissant le plus possible des choix relatifs au cadre de modélisation multi-niveau, de même que les *Design Patterns* du génie logiciel [12] cherchent à s'affranchir du langage objet utilisé. Des *patterns* pour les SMA ont par ailleurs déjà été proposés, notamment des *patterns* d'analyse agent [6] et pour la conception des environnements [22]. Dans cet article, nous cherchons principalement à construire des abstractions destinées à simplifier l'analyse de situations récurrentes pouvant être rencontrées dans divers systèmes multi-niveaux.

2.2 La généralisation cartographique, un problème spatialisé

Les cartes représentent l'information géographique d'une zone donnée de manière d'autant plus simplifiée que l'échelle de la carte est petite (figure 1). Le procédé de simplification, appelé généralisation cartographique, est soumis au respect de contraintes de lisibilité, d'adéquation de la représentation avec le niveau d'abstraction souhaité et de cohérence avec la réalité. La volonté d'automatiser le processus de création de cartes à partir de bases de données géographiques, où la forme des objets est stockée sous forme vectorielle (points, polylignes, polygones), a conduit à la création d'algorithmes permettant d'effectuer cette simplification objet par objet. Néanmoins, les choix des algorithmes, tout comme leur paramétrage, sont autant influencés par l'objet sur lequel ils s'appliquent que par les autres objets en relation (*e.g.* bâtiment à proximité d'un autre, route parallèle à un alignement de bâtiments). Ce constat a motivé l'utilisation de modèles multi-agents pour la généralisation automatisée de cartes.

Le principe de ces modèles multi-agents repose sur la modélisation des objets (*e.g.* bâtiment, tronçon de route, îlot urbain ou pâté de maison)

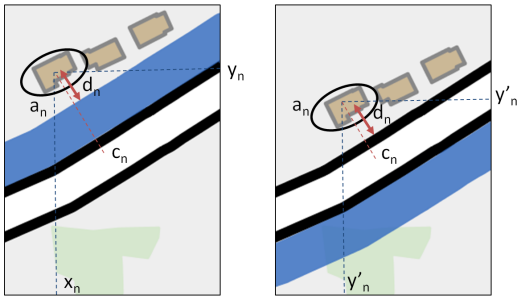


FIGURE 2 – Extrait de carte avec agents bâtiments avec coordonnées cartésiennes et coordonnée en abscisse curviligne du projeté du centroïde sur la route à proximité et distance au projeté. Si le symbole de la route change (par exemple, suite au changement de coté de l'itinéraire de randonnée bleue porté par cette route), alors la position du bâtiment reste la même dans l'environnement de la route, mais change dans celui de la carte.

sous forme d'agents qui cherchent à se généraliser de façon à satisfaire leurs contraintes. Ces contraintes reflètent des critères géométriques de lisibilité et peuvent porter sur un seul objet (e.g. contrainte de taille minimale d'un bâtiment) ou sur plusieurs (e.g. contrainte de non-superposition entre un bâtiment et une route). La construction de l'espace géographique conduit à considérer son organisation selon différents niveaux (e.g. un bâtiment appartient à un îlot urbain, mais peut aussi appartenir à un ou plusieurs alignements de bâtiments ; des points d'intérêts particuliers, comme un refuge ou une table d'orientation, peuvent être situés sur un itinéraire). Plusieurs modèles multi-agents ont été proposés [2, 8, 11, 18, 27, 32], chacun ayant une approche différente des interactions entre niveaux. Nous étudions l'unification de ces modèles [24] en nous appuyant sur le paradigme multi-niveau PADAWAN [21].

2.3 Motifs récurrents en généralisation

Le travail d'unification de modèles en cours nous a conduit à identifier dans un premier temps des motifs récurrents propres à la généralisation automatisée. Nous sommes confrontés à un problème spatialisé impliquant des définitions variées de ce qui peut constituer un groupe d'agents et une multiplicité des aspects multi-niveaux, tant dans la construction des différents niveaux que dans la nature de ces derniers.

En terme de construction de groupes, nous observons plusieurs cas :

- la création d'agents représentant des entités cohérentes d'un point de vue cartographique, et sur lesquelles l'application d'algorithmes particuliers peut s'avérer efficace (e.g. en ville, élimination de bâtiments décidée au niveau de l'îlot urbain pour garder les plus représentatifs la relation meso/composant décrite par [2, 26]) ;
- la division d'un agent, menant à la création d'agents du même type, pour permettre la généralisation de ses sous-parties de manière individuelle, tout en lui laissant un contrôle sur la situation ;
- la création d'agents-points composant la géométrie d'un objet, agents-points pouvant interagir avec d'autres objets de la carte [11] ;
- l'émergence de cas problématiques dont la non-résolution prolongée, voire l'aggravation, peut amener à préférer une résolution à un niveau plus global [9] ;
- des objets dont la position relative à un autre est importante, ce qui nous amène à considérer l'ensemble des ces objets comme appartenant à un même espace ayant un référentiel propre.

Nous observons aussi des types d'environnements différents :

- l'environnement de base, cartésien 2D, où la position des agents est exprimée selon leur coordonnées (x, y) ;
- des environnements relationnels, où les agents identifient leur voisinage à partir d'un graphe de relations, établi *a priori* à partir de l'environnement cartésien, en mesurant la proximité des agents entre eux [8] ;
- des environnements en abscisse curviligne et distance au projeté permettant de positionner un objet particulier vis-à-vis d'un objet linéaire (figure 2) ;
- des environnements décrits de façon à tenir compte de la forme même de cet environnement et de points saillants (e.g. intersections ou virages d'une route) pouvant servir de points de repère [17].

De la diversité de ces situations, nous pensons pouvoir extraire des situations suffisamment génériques. Avant de décrire les *patterns* que nous proposons, nous exposons le cadre formel dans lequel nous les définissons.

3 Cadre formel

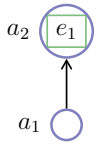
L'approche par *patterns* n'est intéressante que si elle permet au minimum une description uni-

voque et opérationnelle des situations rencontrées, ce qui suppose de disposer d'un cadre formel suffisamment abstrait pour être appliqué à divers contextes. C'est en général UML qui assure cette fonction dans le cas des *patterns* de conception en programmation par objets. Ici, nous proposons d'utiliser une formalisation particulière des relations entre agents (et environnements), que nous illustrons graphiquement. Nous nous appuyons pour cela sur les concepts introduits dans le modèle PADAWAN [21], dont nous reprenons certaines notations. Notre objectif étant toutefois de nous abstraire des choix de modélisation et d'implémentation de l'architecture multi-niveau, nous n'avons conservé de PADAWAN que la part la plus abstraite, susceptible de donner lieu à des réalisations différentes selon les contextes de déploiement visés.

En particulier, la décomposition d'un système complexe multi-niveau dans PADAWAN s'appuient sur deux sortes d'éléments : les agents ($\mathcal{A}$) bien sûr mais également les environnements ($\mathcal{E}$). Ces derniers peuvent désigner indifféremment des « groupes sociaux » ou des portions de l'espace physique, et peuvent eux-mêmes être décrits par des *patterns* [22]. Nous précisons, par souci de s'abstraire de la diversité des implémentations, que nous désignons par environnement un espace doté d'une métrique, dans lequel les agents peuvent se percevoir et interagir. Deux relations fondamentales sont définies entre les agents et les environnements. D'une part, **la situation** d'un agent a dans un environnement e (notée $\mathbf{a} \lhd \mathbf{e}$) exprime le fait que l'agent a peut percevoir, être perçu, agir ou subir des actions dans e. Un agent peut être situé dans un nombre quelconque d'environnements. D'autre part, **l'encapsulation** d'un environnement e par un agent a (notée $\mathbf{a} \sim \mathbf{e}$) signifie que l'agent a « contient » e : il est notamment capable de percevoir tous les agents qui y sont situés et en est **l'hôte**.

À partir de ces deux relations nous définissons la relation **d'hébergement** qui permet de s'affranchir du choix fait dans PADAWAN de multiplier les environnements. Un agent a_1 est **hébergé** par un agent a_2, (ou a_2 est **l'hôte** de a_1), noté $a_1 \sqsubset a_2$ si et seulement si : $\exists e \in \mathcal{E} \mid a_2 \sim e \wedge a_1 \lhd e$. Les relations d'hébergement (avec la situation et l'encapsulation sous-jacentes) sont représentées par un graphe (fig. 3).

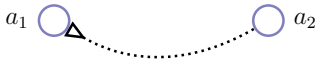
Une autre hypothèse de travail liée au multi-niveau en général est que les comportements d'un agent dépendent de sa situation, autrement dit un agent n'agira pas de la même façon selon



$a_1 \sqsubset a_2$ (a_1 est hébergé par a_2)
i.e. $a_1 \lhd e_1$ (a_1 est situé dans e_1)
et $a_2 \sim e_1$ (a_2 encapsule e_1)

FIGURE 3 – Représentation graphique de la relation d'hébergement

(a) Hébergement optionnel de a_1 par a_2



(b) Dépendance de a_1 vis-à-vis de a_2

FIGURE 4 – Notation des liens d'hébergement optionnel et de dépendance entre agents.

qu'il est hébergé par a_1 ou par a_2. Dans PADAWAN, cela se traduit par le fait que chaque environnement contient une *matrice d'interaction* au sens de l'approche IODA [20] qui définit qui peut faire quoi avec qui. Ici, il suffit de supposer plus généralement que chaque hôte doit spécifier les comportements des agents qu'il héberge. Par ailleurs, ces comportements peuvent s'appuyer sur un certain nombre de primitives, certaines étant directement destinées à assurer des transformations cohérentes des relations d'hébergement. Par exemple dans la suite nous utiliserons **put(a,b)** pour indiquer que l'agent a est désormais hébergé par b ainsi que **remove(a,b)** pour supprimer la relation d'hébergement de a par b ; **del(a)** pour désigner la destruction d'un agent ; et enfin **merge(a,b)** pour noter la fusion de deux agents. Ces primitives, qui ont une définition univoque dans PADAWAN, ont évidemment vocation a être implémentées de façon adéquate dans chaque modèle multi-niveau où l'on souhaite utiliser les patterns que nous avons identifiés.

Outre ces relations, nous notons dans la suite comme indiqué sur la figure 4 deux cas utiles pour la description des patterns : d'une part, un lien d'hébergement *optionnel*, qui indique que l'agent concerné peut ou non être hébergé par l'autre sans affecter la définition du pattern ; d'autre part, un lien de *dépendance*, noté aussi $depends(a_1, a_2)$, qui exprime le fait que l'existence de l'agent a_1 est conditionnée à celle de l'agent a_2 (autrement dit, la suppression de a_2 entraîne par défaut celle de a_1).

4 *Patterns* proposés

L'identification de *patterns* suppose de délimiter des situations univoques, caractérisées par une certaine structuration et d'éventuels effets sur le comportement. Pour chacun des *patterns* proposés, nous décrivons la situation de notre cas d'application qui en est à l'origine, une formalisation et d'autres situations où il peut s'appliquer.

4.1 *Pattern* Agrégation

Origine : Lors de la généralisation automatisée, nous avons besoin, en plus des objets géographiques de base, d'objets qui en sont déduits. Par exemple les villes, qui sont construites par enrichissement des données initiales, par agrégation spatiale de bâtiments proches les uns des autres. Ces agents villes sont appelés à interagir avec les agents qu'ils hébergent : bâtiments, routes, etc ...

Une autre situation est l'identification de groupes constitués d'agents ne réussissant pas à satisfaire leurs contraintes. Dans [9], deux méthodes sont proposées pour identifier les conflits résiduels et créer des entités intermédiaires ayant en charge la résolution de ces conflits.

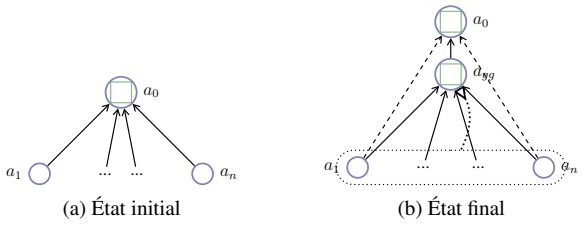
De tels cas nous amènent à considérer le fait qu'un agent construit à partir d'autres agents, peut inclure plusieurs agents différents de ceux qui ont permis sa construction. Nous proposons donc de formaliser cet aspect en introduisant deux fonctions agissant sur un ensemble d'agents :

- une fonction **build :** $\mathcal{A}_{\text{x}} \mapsto a_{\text{gg}}$ qui construit un agent agrégé a_{gg} à partir d'un ensemble d'agents $\mathcal{A}_{\text{x}}$;
- une fonction **populate :** $\mathcal{A}_{\text{x}} \mapsto \mathcal{A}_{\text{agg}}$ qui sélectionne les agents qui seront hébergés par l'agent ainsi créé.

Formalisation : `compose` ($\mathcal{A}_{\text{x}}$, **build, populate) :**

- Prérequis
  - Tous les agents de $\mathcal{A}_{\text{x}}$ sont hébergés pas un même agent.
- Caractérisation :
  $a_{\text{gg}} \leftarrow build(populate(\mathcal{A}_{\text{x}}))$

Cas d'application : cette situation évoque des cas fréquemment rencontrés lors de la simulation à savoir la réification de phénomènes émergents et la création de *clusters*. Ainsi la fonction **build** peut être construite à partir de la fonction d'émergence proposée par [5] ou de la fonction de *clusterisation* proposée par [4].



(a) État initial (b) État final

FIGURE 5 – Notation graphique du pattern *Agrégation*.

Plus généralement, ce pattern permet de réifier par un agent toute structure émergente, tout groupe dans lesquels des agents estiment pouvoir appartenir. Il est à noter que c'est autour de cette problématique de création de niveaux d'abstraction supérieurs que se sont développés les premiers travaux sur la simulation multiniveau, à travers le projet RIVAGE [29] en particulier. Depuis, tout un pan de recherche porte sur la façon d'écrire des fonctions **build** et **populate** de façon générique.

4.2 *Pattern* Décomposition

Origine : Dans le modèle GAEL [11], les objets géographiques considérés comme déformables (*e.g.* courbes de niveaux) ont leur géométrie décomposée en agents points. Dans notre adaptation au sein d'un modèle commun, ces points sont formalisés comme des agents, et sont situés dans un environnement appelé environnement déformable, tout comme les agents points. Ces derniers interagissent de manière à permettre des déformations locales des objets, tout en gardant au mieux leur forme.

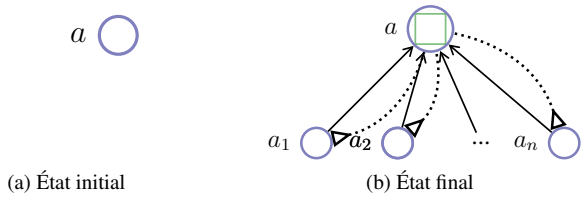
De même, dans le cadre de la généralisation du réseau routier [2], un tronçon de route peut être divisé en sous-tronçons qui seront plus faciles à généraliser individuellement. Néanmoins, afin de maintenir la cohérence de l'ensemble, il est nécessaire que les agents issus de cette subdivision reste liés ensemble. L'agent initial encapsule alors les agents créés, et coordonne leurs actions.

De ces situations, nous constatons que, d'une manière symétrique à l'agrégation, nous pouvons proposer une fonction **build :** $a_x \mapsto \mathcal{A}_{\text{x}}$, pour construire les agents du niveau inférieur. Notons que dans ce cas il n'est pas nécessaire de proposer une fonction **populate** séparée.

Formalisation : **`decompose`** (a, **build**) **:**

— Prérequis : Aucun

— Caractérisation :
$\mathcal{A}_x \leftarrow build(a)$
pour tout $a_i \in \mathcal{A}_x : put(a_i, a)$



(a) État initial (b) État final

FIGURE 6 – Notation graphique du pattern *Décomposition*.

Cas d'application : les cas de décomposition s'effectuent lorsqu'une entité de niveau supérieur doit laisser la place à une entité de niveau moindre. Cela se produit par exemple dans [30], lorsqu'un agent est libéré, ou dans [5], lorsqu'une fonction d'immergence est appelée.

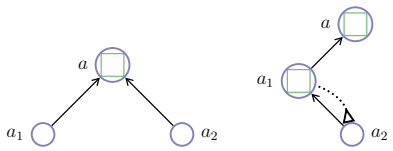
4.3 *Pattern* Hiérarchisation

Origine : Le cas de la hiérarchisation intervient lorsqu'on veut établir une relation hiérarchique entre deux agents. En généralisation, nous pouvons être amenés à définir ce genre de situation lorsqu'un objet semble clairement situé sur un autre objet (e.g. un point d'intérêt comme une table d'orientation indiquée sur le parcours d'un itinéraire décrit sur la carte). Il est important que, suite à la modification de l'objet support, la relation soit préservée, afin que l'information reste cohérente.

Formalisation : **`hierarchise`** (a_1, a_2) **:**

— Prérequis :
  — a_1 et a_2 sont hébergés pas un même agent.

— Caractérisation :
put(a_2,a_1)



(a) État initial (b) État final

FIGURE 7 – Notation graphique du pattern *Hiérarchisation*.

Cas d'application : la hiérarchisation s'applique souvent dans les cas les plus simples où un agent entre dans un autre agent qui se déplace dans le même environnement, comme par exemple un individu qui entre dans un véhicule.

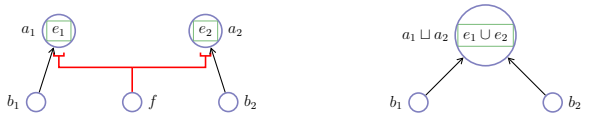
4.4 *Pattern* Frontière

Origine : Dans le cadre de la généralisation, deux îlots urbains adjacents séparés par une route doivent être fusionnés lorsque cette route séparatrice disparaît pour une raison tierce, comme la volonté de diminuer la densité du réseau routier dans une ville. Le résultat de cette fusion, un nouvel îlot, doit héberger l'ensemble des bâtiments qui étaient hébergés dans les îlots initiaux. Dans [1], cette situation a pu être intégrée par l'introduction de la notion de frontière. Nous parlons de frontière lorsque deux environnements sont séparés par un objet appartenant aux deux environnements. Le rôle de la frontière est d'interdire le passage d'agents d'un environnement à l'autre.

`frontiere` **(f, { a$_1$, a$_2$ }) :**

— Prérequis
  — $f \sqsubset a_1$ et $f \sqsubset a_2$
  — a$_1$ et a$_2$ sont des agents « compatibles » (on peut les « fusionner »).

— Caractérisation : **`del`** **(f)** $\rightarrow$ **`merge`** **(a$_1$, a$_2$)**



(a) f est un agent frontière (b) Après suppression de f

FIGURE 8 – Notation et résultat de la suppression d'un agent frontière.

Cas d'application : l'agent frontière peut être utilisé dans les situations où des environnements sont séparés par un élément dont le dynamisme (qui peut, comme dans notre exemple, se limiter à sa disparition) justifie sa modélisation sous forme d'agent.

Nous pouvons nous interroger ici sur le fait que, dans les exemples proposés, la frontière est hermétique. Or, certaines situations impliquent une certaine porosité, comme la modélisation de la paroi d'une cellule, ou la modélisation d'un sas. Pour tenir compte de ce genre de situation, nous proposons un autre *pattern* spécifique : porte.

4.5 *Pattern* Porte

Origine : Nous ne sommes pas confronté à cette situation dans la cadre de la généralisation,

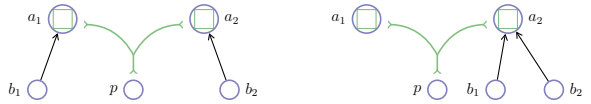
mais la caractérisation du pattern frontière nous amène à identifier un *pattern* porte.

`porte` (p, { a_1, a_2 }) :

— Prérequis :

— $p \sqsubset a_1$ et $p \sqsubset a_2$

— Caractérisation : Le pattern porte permet de définir une primitive **cross(a,p)** qui permet à un agent a situé dans a_1 (resp. a_2) de demander à l'agent porte p de le déplacer vers a_2 (resp. a_1). Cette primitive peut être assujettie à des conditions propres à la porte et aux agents hôtes concernés. La suppression d'une porte n'a pas d'effets sur les hôtes.



(a) p est un agent porte (b) b_1 a traversé la porte p

FIGURE 9 – Notation et traversée d'un agent porte.

Cas d'application : un exemple d'application du modèle PADAWAN [21], la pompe sodium-potassium, peut être modélisé selon ce *pattern* porte. La simulation d'un système impliquant un sas est un autre exemple.

4.6 *Pattern* Embarquement

Origine : En généralisation, lorsque nous modifions la géométrie ou le symbole d'un objet, il est souhaité que les relations que cet objet possède par ailleurs avec d'autres objets soient préservées. Par exemple dans [23], une impasse peut glisser le long de la route la supportant afin de laisser de la place aux bâtiments. Si cela se produit les bâtiments dans son voisinage doivent être déplacés afin de garder leur position relative par rapport à l'impasse. De même (figure 2), les objets à proximité d'une route doivent garder une position relative identique lorsque la route change de symbole, par exemple lorsque le symbole d'un itinéraire change de position.

Pour résoudre cette situation dans le cadre de PADAWAN, nous nous appuyons sur la notion d'« environnement embarqué ». Lorsqu'un agent possède, dans son voisinage, un ensemble d'objets dont la position relative par rapport à lui est importante, un environnement embarqué est créé, ainsi qu'un agent encapsulant cet environnement. Cet environnement embarque à la fois l'agent à l'origine de sa création, et les agents de son voisinage.

L'environnement embarqué définit son propre référentiel, qui s'appuie sur l'agent à l'origine de sa création, et dans lequel les autres agents ont des coordonnées relatives. Ainsi, si l'agent à l'origine de l'embarquement se voit modifié, la relation entre le référentiel de l'environnement embarqué et le référentiel de l'environnement dans lequel il se situait initialement change aussi. Par conséquent, comme les agents doivent préserver leurs positions relatives dans l'environnement embarqué, ils devront modifier leur position dans l'environnement où ils se situaient auparavant.

Dans le cadre de nos exemples, un référentiel simple consiste à utiliser des coordonnées composées de l'abscisse curviligne du projeté du centroïde de l'agent embarqué, et de sa distance à l'objet linéaire (route) à l'origine de l'embarquement. Lors de modifications de l'objet linéaire, le référentiel local est modifié par rapport au référentiel cartésien de la carte, ce qui implique une modification de la position des objets de l'environnement embarqué dans le référentiel de la carte.

`embarque` ($a^\star$, $\mathcal{A}_x$) :

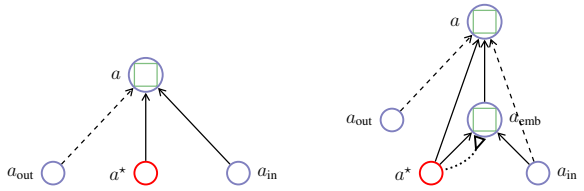
— Prérequis : $a^\star$ identifié comme jouant un rôle clef pour le référencement spatial dans son voisinage $a^\star$, $\mathcal{A}_x$.

— Caractérisation :

  — Création de l'environnement embarqué e_{emb} et de l'agent l'hébergeant : c'est un cas particulier d'agrégation, où l'agent à l'origine de l'embarquement a une relation de dépendance avec l'agent agrégat (encapsulant l'environnement embarqué) créé.

  — modification de $a^\star \rightarrow$ mise à jour de la position des agents $a \in \mathcal{A}_x$.

  — modification de la position dans l'environnement initial d'un agent $a \in \mathcal{A}_x \rightarrow$ mise à jour des coordonnées de a.



(a) État initial (b) État initial

FIGURE 10 – Notation graphique du *pattern Environnement embarqué*.

Cas d'application : Plusieurs situations de simulation peuvent être modélisées selon la no-

tion d'agent embarqué. Par exemple, une simulation de déplacement d'astres peut modéliser leur champs de gravitation. Dans une telle modélisation, la Terre définirait un environnement embarqué dans lequel se situerait la Lune. Le déplacement de la Terre autour du Soleil impliquerait une modification de la position de la Lune dans le référentiel héliocentrique.

4.7 *Patterns* Meso

Les interactions hiérarchiques, c'est-à-dire impliquant un hôte et ses hébergés, sont particulières, dans la mesure où elles peuvent impliquer des prises de décisions fortes d'un niveau supérieur pour les agents hébergés.

Origines : Dans le modèle AGENT, un *meso* désigne un agent qui est composé d'autres agents (ces dernier pouvant être soit *micro*, c'est-à-dire des agents existant initialement, soit *meso* eux aussi). Les *mesos* ont plusieurs rôles vis-à-vis des autres agents. Plusieurs de ces rôles sont décrits dans [26] : le **Législateur**, le **Contrôleur** et le **Coordinateur**. Le **Coordinateur** décrit une situation particulière qui fait l'objet d'un *pattern* Ordonnanceur décrit à part.

Le **Législateur** agit directement sur les connaissances des agents qu'il héberge, ainsi que leurs motivations. Dans le cadre de la généralisation, cela se traduit par le fait que le *meso* peut modifier directement les contraintes des ses composants, et ainsi influer leur comportement. Le **Contrôleur** assure que les actions effectuées par les agents vont dans le sens de ses intérêts (dans le cadre de la généralisation, dans le sens de la satisfaction de ses contraintes, mais aussi de celle de ses composants). Il peut décider d'annuler une action effectuée par un de ses agents hébergés. Dans [18], les agents hébergés sont amenés à interagir eux-mêmes pour se généraliser, mais peuvent parfois s'en remettre à un agent hôte pour décider d'une action qu'ils ne peuvent résoudre eux même. Nous proposons la dénomination d'**Arbitre** pour ce cas de figure. Dans [25], les agents bâtiments appartenant à un alignement de bâtiments peuvent décider de s'en remettre à l'agent alignement pour effectuer une action qui sera mieux effectuée par ce dernier. Nous proposons ici le terme de **Délégataire** pour décrire ce rôle de *meso*.

Cas d'application : Dans le cadre d'une simulation avec un groupe hiérarchisé, comme un orchestre où le chef choisit l'ordre d'activation des membres du groupe, nous pouvons modéliser des situations impliquant certains des rôles du meso.

4.8 *Pattern* Ordonnanceur

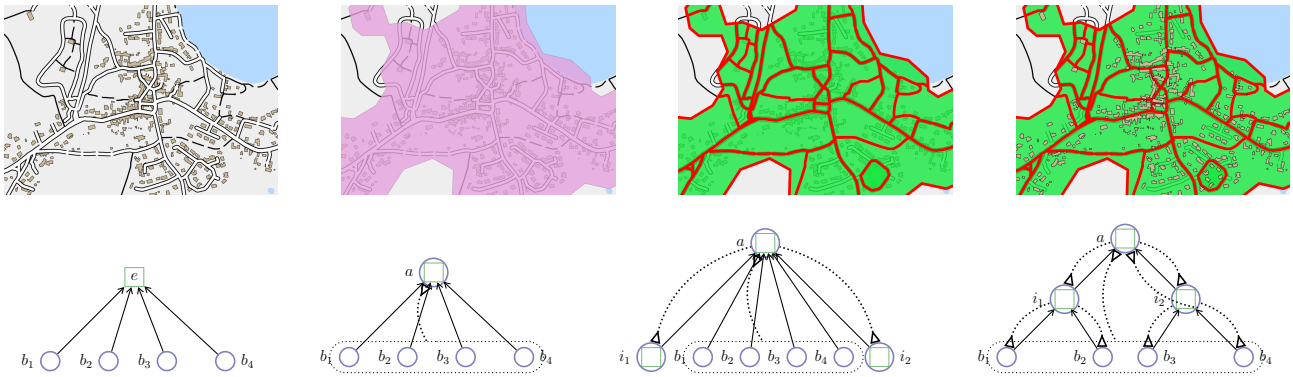
Origine : Dans le modèle AGENT, les agents meso ont la possibilité d'activer leurs composants, afin qu'ils se généralisent eux-même. Cette activation s'effectue lors du cycle de vie de l'agent meso. Ce rôle d'activation appelé **Coordinateur** dans [26], permet d'identifier le meilleurs moment pour la généralisation individuelle des bâtiments. Les connaissances qu'ont les îlots de l'intégralité de ces composants en font les agents les plus appropriés identifier l'ordre d'activation le plus opportun.

Cas d'application : Dans [13], une approche pour la simulation propose de donner aux holons un rôle équivalent à celui de Coordinateur.

5 Discussion

Les patterns proposés en section 4 sont des premières propositions pour l'identification de patterns. Ces *patterns* permettent l'analyse et la comparaison de situation récurrentes. Leur intérêt réside aussi dans la combinaison de ces *patterns*, et dans leur utilisation lors de la conception de modèle. Par exemple, plusieurs situations particulières propres à la généralisation cartographique peuvent être reproduites à partir d'une combinaison de ces *patterns*. Ainsi, la création d'îlots urbains [3], et leur relation avec les bâtiments qu'ils hébergent, peut s'exprimer à l'aide des trois *patterns* de création de niveau utilisés séquentiellement, comme montré dans la figure 11. Ensuite, à partir des agents îlots ainsi obtenus, des relations hiérarchiques spécifiques sont appliquées : ce sont les relations de Contrôleur et de Législateur décrites lors de la définition du pattern meso. Ainsi les *patterns* de création de niveau peuvent s'accompagner d'autres *patterns* spécifiques. L'utilisation conjointe de *patterns* est aussi pertinente en dehors de la généralisation. Par exemple, la modélisation d'une membrane cellulaire nécessite l'utilisation conjointe des *patterns* frontière et porte.

Outre qu'ils peuvent être utilisés conjointement, nous constatons aussi que les *patterns* proposés peuvent présenter des liens. Ainsi le *pattern* Agrégation et le *pattern* Embarquement présentent des similitudes. Néanmoins, les spécifi-



État initial : les bâtiments b_i sont situés dans un même environnement

Étape 1 : **agrégation** des bâtiments b_i en une ville a

Étape 2 : **décomposition** de la ville a en îlots i_k

Étape 3 : **hiérarchisation** des bâtiments au sein de leurs îlots respectifs

FIGURE 11 – Explicitation de la construction d'îlots urbains proposée par [3] à l'aide de *patterns*.

cités propres à l'agent embarqué, et la définition de comportement de maintien de la position relative qui motive sa création justifient sa définition de manière indépendante.

En termes d'exhaustivité, les *patterns* proposés sont issus des recherches effectués dans le cadre de la généralisation s'appuyant sur des modèles orientés agents. Une des contraintes de la généralisation étant de préserver relativement la position des objets sur la carte, le procédé n'implique pas une grande mobilité des agents. Il est donc très probable que les *patterns* pouvant être extraits des SMA pour la généralisation cartographique ne constituent pas une liste exhaustive, et que de nombreux *patterns* multi-niveaux intéressants restent à être identifié.

En revanche, en termes de généricité nous avons cherché à nous affranchir autant que possible du modèle d'origine dans lequel les *patterns* proposés ont été mis en œuvre, afin de ne retenir que les caractéristiques les plus générale d'une décomposition multi-niveau.

6 Conclusion et perspectives

Nous avons proposé un début de formalisation de *patterns* multi-niveaux pour des SMA. Ces *patterns*, identifiés à partir de cas particulier rencontrés lors de la mise en commun de modèles pour la généralisation cartographique, portent sur trois aspects spécifiques de relations multi-niveaux : la création de niveaux différents, des aspects structurels de la relation entre niveaux et les relations spécifiques d'un agent avec un agent d'un niveau supérieur dans lequel il est hébergé. Ces *patterns* d'analyse ont, à moyen terme, vocation à être étendus pour servir également à la conception de systèmes multi-niveaux. La liste des *patterns* proposés n'est pas exhaustives et gagnerait à être enrichie de nouvelles situations, à identifier à partir d'autres SMA multi-niveaux.

Références

[1] A. Atrash. Adaptation of PADAWAN multi-agent model for the solution of a spatial problem : the cartographic generalization. Rapport de stage, Univ. Paris Sud, 2011.

[2] M. Barrault, N. Regnauld, C. Duchêne, K. Haire, C. Baeijs, Y. Demazeau, P. Hardy, W. Mackaness, A. Ruas, et R. Weibel. Integrating multi-agent, object-oriented, and algorithmic techniques for improved automated map generalization. In *20th International Cartographic Conference (ICC'01)*, vol. 3, p. 2110—2116. International Cartographic Association, 2001.

[3] A. Boffet. Analyse multi-niveaux des espaces urbains. *Revue Internationale de Géomatique*, 12(2) :215–260, 2002.

[4] P. Caillou et J. Gil-Quijano. Description automatique de dynamiques de groupes dans des simulations à base d'agents. In P. Chevaillier et B. Mermet, éds., *Actes des 20e Journées Francophones sur les Systèmes Multi-Agents (JFSMA'2012)*, p. 23–32. Cépaduès, 2012.

[5] B. Camus, J. Siebert, C. Bourjot, et V. Chevrier. Modélisation multi-niveaux dans AA4MM. In P. Chevaillier et B. Mermet, éds., *Actes des 20e Journées Francophones sur les Systèmes Multi-Agents (JFSMA'2012)*, p. 43–52. Cépaduès, 2012.

[6] V. Couturier, D. Telisson, et M.-P. Huget. Patterns d'analyse pour l'ingénierie des systèmes multi-agents. In M. Occello et L. Rejeb, éds., *Actes des 18e Journées Francophones sur les Systèmes Multi-Agents (JFSMA'2010)*, p. 55–64. Cépaduès, 2010.

[7] D. David, D. Payet, et R. Courdier. Réification de zones urbaines émergents dans un modèle simulant l'évolution de la population à La Réunion. In

E. Adam et J.-P. Sansonnet, éds., *Actes des 19e Journées Francophones sur les Systèmes Multi-Agents (JFSMA'2011)*, p. 63–72. Cépaduès, 2011.

[8] C. Duchêne, A. Ruas, et C. Cambier. The CARTACOM model : transforming cartographic features into communicating agents for cartographic generalisation. *International Journal of Geographical Information Science*, 26(9) :1533–1562, 2012.

[9] C. Duchêne et G. Touya. Emergence de zones conflits dans deux modèles de généralisation cartographique multi-agents. In M. Occello et L. Rejeb, éds., *Actes des 18e Journées Francophones sur les Systèmes Multi-Agents (JFSMA'2010)*, p. 33–42. Cépaduès, 2010.

[10] J. Ferber, F. Michel, et J. Báez. AGRE : Integrating environments with organizations. *LNCS*, vol. 3374, p. 48–56. Springer, 2005.

[11] J. Gaffuri, C. Duchêne, et A. Ruas. Object-field relationships modelling in an agent-based generalisation model. In *Workshop on generalisation and multiple representation*, 2008.

[12] E. Gamma, R. Helm, R. Johnson, et J. Vlissides. *Design Patterns, Elements of Reusable Object-Oriented Software*. Addison Wesley, 1994.

[13] N. Gaud, S. Galland, et A. Koukam. Towards a multilevel simulation approach based on holonic multiagent systems. In *Computer Modeling and Simulation. UKSIM 2008. Tenth International Conference on*, p. 180–185. IEEE, 2008.

[14] J. Gil-Quijano, G. Hutzler, et T. Louail. De la cellule biologique à la cellule urbaine : retour sur trois expériences de modélisation multi-échelles à base d'agents. In Z. Guessoum et S. Hassas, éds., *Actes des 17e Journées Francophones sur les Systèmes Multi-Agents (JFSMA'2009)*, p. 187–198. Cépaduès, 2009.

[15] N. Hamani, J.-P. Jamont, M. Occello, et M. Koudil. Ant-MWAC : Une approche conjointe multi-agent et colonie de fourmis pour gérer les communications dans les réseaux de capteurs sans fil. In E. Adam et J.-P. Sansonnet, éds., *Actes des 19e Journées Francophones sur les Systèmes Multi-Agents (JFSMA'2011)*, p. 85–94. Cépaduès, 2011.

[16] T.-T.-H. Hoang, M. Occello, et J.-P. Jamont. Un modèle multi-agent générique récursif pour simplifier la supervision de systèmes décentralisés multi-niveaux. In E. Adam et J.-P. Sansonnet, éds., *Actes des 19e Journées Francophones sur les Systèmes Multi-Agents (JFSMA'2011)*, p. 41–50. Cépaduès, 2011.

[17] K. Jaara, C. Duchêne, et A. Ruas. Preservation and modification of relations between thematic and topographic data throughout thematic data migration process. In M. Buchroithner et al., éds., *Sel. Contrib. to the 26th Int. Conf. of the ICA*, Lecture Notes in Geoinformation and Cartography, p. 103–118. Springer, 2013.

[18] N. Jabeur, B. Boulekrouche, et B. Moulin. Using multiagent systems to improve real-time map generation. In *Advances in Artificial Intelligence*, p. 37–48. Springer, 2006.

[19] Y. Kubera, P. Mathieu, et S. Picault. Everything can be agent ! In W. van der Hoek et al., éds., *9th Int. Joint Conf. on Auton. Agents and Multi-Agent Systems (AAMAS)*, p. 1547–1548, 2010.

[20] Y. Kubera, P. Mathieu, et S. Picault. IODA : An interaction-oriented approach for multi-agent based simulations. *J. Auton. Agents and Multi-Agent Systems (JAAMAS)*, 23(3) :303–343, 2011.

[21] P. Mathieu et S. Picault. An interaction-oriented model for multi-scale simulation. In T. Walsh, éd., *Proc. of the 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI'2011)*, p. 332–337. AAAI, 2011.

[22] P. Mathieu, S. Picault, et Y. Secq. Les environnements : en avoir ou pas ? Formalisation du concept et patterns d'implémentation. In R. Courdier et J.-P. Jamont, éds., *Actes des 22e Journées Francophones sur les Systèmes Multi-Agents (JFSMA'2014)*, p. 55–54. Cépaduès, 2014.

[23] A. Maudet, G. Touya, C. Duchêne, et S. Picault. Improving multi-level interactions modelling in a multi-agent generalisation model : first experiments. In D. Burghardt, éd., *Proc. of the 16th ICA Workshop on Generalisation and Map Production*, 2013.

[24] A. Maudet, G. Touya, C. Duchêne, et S. Picault. Representation of interactions in a multi-level multi-agent model for cartography constraint solving. In Yves Demazeau et al., éds., *Proc. of the 12th Int. Conf. on Practical Applications of Agents and Multi-Agent Systems (PAAMS'2014)*, no. 8473 in LNCS, p. 183–194. Springer, 2014.

[25] J. Renard. Introduction de structures réactionnelles à activation ascendante dans une organisation hiérarchique descendante d'agents - application à la généralisation des alignements urbains. In P. Chevaillier et B. Mermet, éds., *Actes des 20e Journées Francophones sur les Systèmes Multi-Agents (JFSMA'2012)*, p. 139–149. Cépaduès, 2012.

[26] A. Ruas. The role of meso objects for generalisation. In *9th Int. Symposium on Spatial Data Handling (SDH'00)*, Beijing (China), 2000.

[27] M. N. Sabo, Y. Bédard, B. Moulin, et E. Bernier. Toward self-generalizing objects and on-the-fly map generalization. p. 155–173, 2008.

[28] L. Sanders, D. Pumain, H. Mathian, F. Guérin-Pace, et S. Bura. SIMPOP : a multi-agents system for the study of urbanism. *Environment and Planning B*, 24 :287–305, 1997.

[29] D. Servat, E. Perrier, J.-P. Treuil, et A. Drogoul. When agents emerge from agents : Introducing multi-scale viewpoints in multi-agent simulations. *LNCS*, vol. 1534, p. 183–198, Paris, France, 1998. Springer.

[30] D. A. Vo, A. Drogoul, et J.-D. Zucker. An operational meta-model for handling multiple scales in agent-based simulations. In *Proc. of the IEEE Int. Conf. on Computing & Communication Technologies, Research, Innovation, and Vision for the Future (RIVF)*. IEEE Press, 2012.

[31] D. Weyns, H.V.D. Parunak, F. Michel, T. Holvoet, et J. Ferber. Environments for multiagent systems. State-of-the-Art and research challenges. *LNCS*, vol. 3374, p. 1–47. Springer, 2005.

[32] X. Zhang et E. Guilbert. A multi-agent system approach for feature-driven generalization of isobathymetric line. In *Advances in Cartography and GIScience. Volume 1*, p. 477–495. Springer, 2011.