Formalismes de description des modèles agent

Fabrice Bouquet, David Sheeren, Nicolas Becu, Benoit Gaudou, Christophe Lang, Nicolas Marilleau, Claude Monteil

Chapitre 2

Formalismes de description des modèles agent

2.1. Introduction

Ce chapitre a pour but de présenter les bonnes pratiques et l'apport de la formalisation dans le domaine de la modélisation de systèmes multi-agents (SMA). Pour cela, les auteurs rappellent dans un premier temps l'intérêt de modéliser des systèmes en mettant en perspective les paradigmes associés à la démarche multi-agents. Il est alors argumenté que l'utilisation des langages de modélisation graphique permettent un meilleur échange entre les partenaires intervenant dans la conception d'un SMA. Deux types de modèles graphiques reposant sur un même socle sémantique sont ensuite présentés : UML (*Unified Modeling Language*) et AML (*Agent Modeling Language*). Le premier a une vocation généraliste et permet de faire une analyse de l'ontologie et des dynamiques du système modélisé. Le second mobilise les paradigmes particuliers aux agents et permet de faire une conception plus proche du SMA qui sera produit. Après avoir discuté de l'intérêt de chacune de ces modélisations graphiques et présenté quelques extensions possibles, le chapitre aborde l'intérêt et la façon de documenter un modèle multi-agents. Pour ce faire, le protocole ODD (*Overview, Design concepts, Details*) qui guide le modélisateur dans la rédaction d'une documentation des objectifs, des constituants et des propriétés spécifiques du modèle, est présenté.

Nous illustrons chacun des concepts présentés (UML, AML et ODD) au travers de leur application à un exemple qui servira de fil rouge dans le reste de cet ouvrage.

Chapitre rédigé par F. BOUQUET, D. SHEEREN, N. BECU, B. GAUDOU, C. LANG, N. MARILLEAU et C. MONTEIL.

2.2. Exemple fil rouge

Pléthore de cas d'application mettent en évidence l'intérêt de l'approche agent pour modéliser des phénomènes complexes. Ils concernent de nombreux domaines comme par exemple l'écologie, les sciences sociales ou l'épidémiologie. Nous avons choisi d'aborder le domaine de l'épidémiologie car d'une part ce thème embrasse plusieurs autres domaines notamment l'écologie et les sciences sociales, et d'autre part, une grande variété des concepts multi-agents peut être abordée pour modéliser de tels phénomènes complexes.

Ainsi, ce chapitre et les suivants seront illustrés par un même exemple fil rouge de modélisation d'un phénomène épidémiologique. Il s'intéresse à la dispersion géographique d'une épidémie transmise par le moustique à l'homme. L'enjeu est de comprendre et mesurer l'impact des mobilités pendulaires des hommes (mobilité domicile $\leftrightarrow$ travail) sur le développement et la propagation d'une maladie contagieuse.

Nous avons choisi de nous intéresser au paludisme, maladie présente dans nombre des pays d'Afrique notamment au Cameroun et dans la sous-région du Maroua (voir figure 2.1).

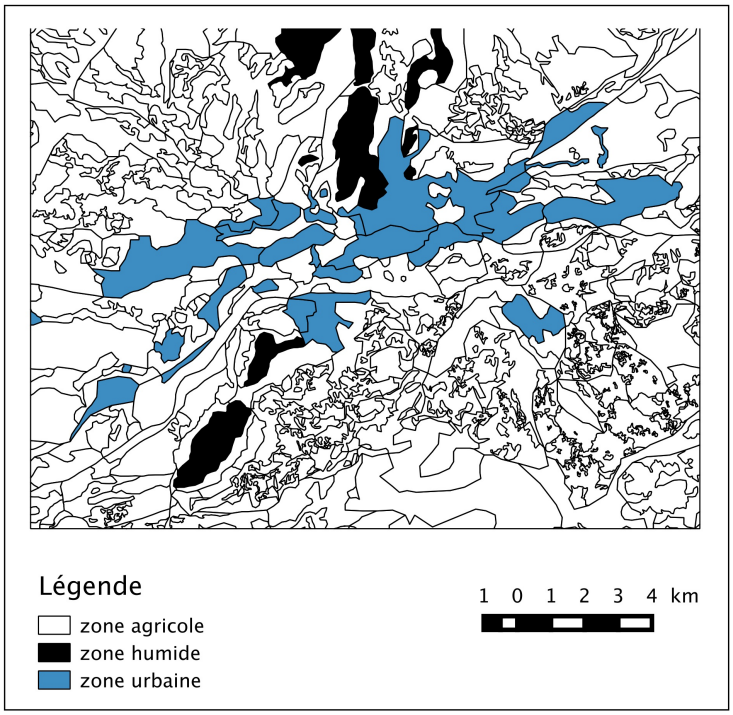

Figure 2.1. *Carte de la sous région du Maroua (Cameroun)*

Le paludisme est transmis à l'homme uniquement par des piqûres de moustiques Anophèles qui ont été auparavant infectés. Un moustique sain devient à son tour infecté lorsqu'il pique un homme infecté. Il n'y a pas de contamination homme-homme ou moustique-moustique. Par ses propres moyens, le moustique a un rayon de déplacement très faible de l'ordre de 50m par jour mais demeure présent sur la totalité du territoire. Néanmoins le véritable vecteur de dispersion de l'épidémie semble être l'homme dans la mesure où il doit se déplacer sur de grandes distances pour accomplir ses activités quotidiennes.

La majorité des habitants de la sous-région du Maroua réside dans les zones urbanisées (en noir dans la figure 2.1). Les cultivateurs de la région réalisent quotidiennement une mobilité pendulaire entre leur domicile (en ville) et leur zone de culture, à l'extérieur de la ville (zones blanches et bleues dans la figure 2.1). Les zones bleues correspondent aux zones humides dans lesquelles les moustiques peuvent pondre leurs œufs.

Dans la suite, ce système sera modélisé en utilisant le langage UML (section 2.3.1), puis le langage AML (section 2.3.2) et documenté à l'aide du protocole ODD (section 2.4). Le lecteur pourra ainsi découvrir par l'exemple les différentes formes de description de modèle.

2.3. Formalisation de modèles agents

Dans cette partie, nous précisons d'abord les finalités de la formalisation de systèmes multi-agents, puis présentons deux outils destinés à faciliter cette formalisation – les langages UML et AML – qui sont illustrés sur l'exemple fil rouge.

La formalisation est une démarche qui a trois finalités. La première concerne l'analyse du système à étudier. Il faut effectivement un outil apte à aider à la compréhension du système visé. La seconde concerne l'abstraction qui nous permet de ne pas être contraint par des considérations techniques liées aux éléments de simulation. La troisième est la génération de code, facilitant ainsi le passage du modèle à l'implémentation. Par ailleurs les formalisations graphiques comme UML et AML permettent également de fluidifier la communication entre plusieurs personnes à propos du contenu d'un modèle. Elles ont l'avantage de pouvoir représenter de manière synthétique par un ou plusieurs diagrammes, des structures et des mécanismes complexes. L'aspect graphique des diagrammes est un élément simplificateur pour leur compréhension par des non-programmeurs.

Nous choisissons, dans cet ouvrage, de nous concentrer sur UML et AML, ce dernier constituant une extension du premier plus spécialement dédiée au paradigme agent en formalisant la description des comportements d'agents et de leurs interactions.

Les différents diagrammes hérités d'UML peuvent être aussi naturellement utilisés dans le cadre du développement des modèles. Sa généricité pose toutefois certains problèmes quand il s'agit de l'utiliser dans un contexte particulier. Ainsi, dans le cas des simulations spatialisées, il faudra associer un langage apte à représenter l'espace et ses contraintes, comme les représentations sous-jacentes des SIG (Systèmes d'Information Géographiques) [CHI 13].

2.3.1. *Le langage UML (*Unified Modeling Language*)*

Dans les années 1970 et 1980 planait un désaccord entre les personnes qui croyaient en la modélisation de données et ceux qui croyaient en la modélisation fonctionnelle. Á cette époque, les utilisations de diagramme de flux et de diagramme relationnel étaient généralement considérées comme mutuellement exclusives.

Ces deux camps se sont réconciliés vers la fin des années 1980 et ont pris conscience que la plupart des projets pouvaient bénéficier des deux types de modèles. Cette réconciliation a été suivie de l'apparition de nombreuses méthodes d'analyse et de modélisation orientées objet. Toutefois chaque méthode possédait sa propre notation et ses définitions de termes comme objet, type et classe. Il n'y avait aucun standard. Le nombre de langages de modélisation a augmenté de moins de dix à plus de cinquante entre 1989 et 1994.

Dans la fin de l'année 1994, Grady Booch et Jim Rumbaugh ont annoncé leur collaboration dans l'élaboration d'une *Méthode unifiée*. Ils ont été rejoints un peu plus tard par Ivar Jacobson. Finalement, après quelques années d'expérimentation avec les différentes notations et différents concepts, le groupe s'est basé sur quelques-uns d'entre eux pour débuter l'établissement de sémantiques pour les concepts orientés objet et se mettre d'accord sur une notation commune.

Pendant l'année 1996, l'*Unified Method* a évolué en *Unified Modeling Language* (UML). Ce nouveau nom était destiné à souligner le fait qu'UML était un langage de modélisation et non une méthode. Son but était de fournir une notation expressive, de définir une sémantique pour les concepts impliqués et de laisser le choix du processus de développement. Finalement, né de la fusion des trois méthodes de modélisation objet OMT, Booch et OOSE, UML est devenu en quelques années un standard incontournable. Créé à l'origine pour permettre au développeur de représenter, spécifier, analyser et visualiser la structure d'un projet en programmation orientée objet, UML est aujourd'hui utilisé dans un grand nombre de domaines.

L'évolution d'UML est analogue à l'évolution des logiciels : il y a des versions *majeures* ainsi que des améliorations et des extensions. La version actuelle d'UML est la version 2.0 [UML05] qui est décomposée en quatre parties :

– **UML 2.0 Superstructure :** les diagrammes utilisés pour la modélisation ;

– **UML 2.0 Infrastructure :** fondations partagées avec MOF 2.0 ;

– **UML 2.0 OCL :** le langage de contraintes ;

– **UML 2.0 Diagram Interchange :** permet l'échange de diagrammes (y compris l'aspect graphique) entre outils.

UML Superstructure contient différents types de diagrammes :

– **6 diagrammes de structure :** classes, objets, structure composite, composants, déploiement, package ;

– **3 diagrammes de comportement :** activités, cas d'utilisation, machines d'états ;

– **4 diagrammes d'interaction :** séquence, communication, vue d'ensemble des interactions, timing.

2.3.1.1. *Formalisation de la structure des modèles (diagrammes statiques)*

Cette partie présente l'aspect descriptif de la modélisation UML. Elle est aussi appelée partie statique ou de structure. Nous ne présenterons ici que deux des six diagrammes de structure : le diagramme de classes et le diagramme d'objets. Ils correspondent aux diagrammes les plus utilisés lors de la modélisation UML. C'est aussi le diagramme de classes qui est utilisé pour les aspects de méta-modèles (modèle de modèles) encore appelés ontologie.

2.3.1.1.1. Diagramme de classes

Le diagramme de classes UML permet de modéliser la structure, la partie statique d'un système. Les classes sont essentielles car elles définissent un type abstrait qui permettra plus tard d'instancier des objets dans le diagramme d'objets. La figure 2.2 présente le diagramme de classes complet de notre exemple fil rouge. On peut voir la classe `Entité`. Cette classe possède un attribut `est-infecté` de type booléen et une fonction `infecter()` qui permet d'infecter une autre `Entité`. Par ailleurs, il s'agit d'une entité située ; elle possède donc un attribut `positionActuelle` qui est le lieu sur lequel elle se trouve, et la méthode `déplacer`. Les classes ne sont pas isolées dans ce diagramme. Elles peuvent avoir des liens entre elles appelés relation. On peut avoir des relations entre des classes pour différentes raisons :

– *l'héritage* permet à une classe d'hériter de l'ensemble des attributs et méthodes de la classe mère dont elle hérite. Cette relation est représentée par une flèche pleine. Ainsi les classes `Moustique` et `Homme` héritent notamment de l'attribut `est-infecté` de la classe `Entité`. Outre les attributs et méthodes hérités, la classe `Moustique` possède également l'attribut `patient0`. Celui-ci représente les moustiques qui sont contaminés à l'initialisation de la simulation appelé patient zéro. Il est représenté par un booléen qui est vrai dans ce cas et faux sinon.

– *l'association* permet de donner un lien entre deux classes. Elle peut être nommée et contenir des informations de multiplicité (cardinalité) et de navigation (sens de

la relation). Dans l'exemple, nous avons représenté l'association `A contaminée` dont la connexion est particulière puisqu'elle relie la classe `Entité` et elle-même (association réflexive) pour indiquer la chaine de contamination indiquant qui `contamine` et l'entité `infectée`. La *cardinalité* `"*"` indique qu'une entité peut être le contamineur de 0 ou plusieurs entité et la cardinalité `"0..1"` à l'autre bout de l'association indique qu'une `entité` a été infectée par zéro ou un seul contamineur. Il existe deux cas particuliers d'association qui sont l'*agrégation* et la *composition* représentées respectivement par un losange évidé ou plein du côté de l'agrégat. Dans notre exemple, une relation de composition relie les lieux (la classe `Lieu`) au `Territoire` (agrégat).

2.3.1.1.2. Représentation de l'exemple fil rouge avec UML

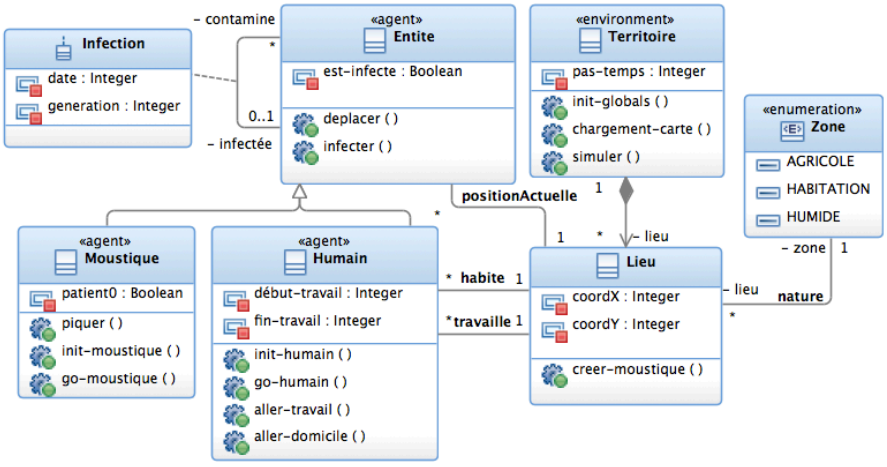

Figure 2.2. *Représentation du diagramme de classes de l'exemple fil rouge*

Dans le modèle UML, nous avons regroupé les éléments communs des entités mobiles que sont les moustiques et les humains dans une classe `Entité`. Nous représentons la notion de contamination entre les entités par une association dotée d'attributs nommée «Infection». L'information n'apparaît pas dans le diagramme (car nous n'avons pas fait apparaitre les contraintes) mais une entité ne peut être infectée que par une entité d'un sous-type différent. Pour cela, on garde l'information concernant la source à travers l'association mais aussi la date et la génération à travers les attributs de l'association.

Le `Territoire` est composé d'éléments de la classe `Lieu` qui correspondent aux différentes zones d'occupation du sol possibles, précisées grâce à l'attribut «nature» dans la classe `Lieu`. Il s'agit d'un attribut de type énuméré dont les différentes modalités sont précisées dans la classe `Zone`. Par ailleurs, dans notre représentation, le `Territoire` joue également le rôle d'environnement du système et il contient la

représentation du temps (`pas-temps`) dans la simulation et toute la partie gestion du système. Nous retrouvons les éléments de gestion de la simulation pour démarrer cette dernière après avoir initialisé les différents composants et chargé la carte. Á noter qu'un `Lieu` est représenté par ses coordonnées représentant le centre de la zone. Initialement, il faut pouvoir créer des moustiques infectés. Cela est représenté par la méthode `créer-moustique`.

2.3.1.1.3. Diagramme d'objets

Le diagramme d'objets permet d'instancier les classes définies dans le diagramme de classes en objets réels. Ce diagramme est utile pour donner une image de l'état du système à un instant t. Pour modéliser l'état initial d'un système par exemple, un diagramme d'objets est utilisé. Dans la figure 2.3, nous avons représenté 3 entités dont 2 moustiques (un patient zéro et un non contaminé) et un humain. À l'instant t la chaîne de contamination se limite au patient zéro. Il est à noter la possibilité de donner pour tout ou partie des attributs des valeurs pour l'objet. Dans l'exemple, nous n'avons pas représenté l'ensemble des objets et relations. Par exemple, les agents sont situés et ils devraient être en relation avec des instances de la classe `Lieu` et nous n'en avons représenté qu'une seule `Lieu3-7`.

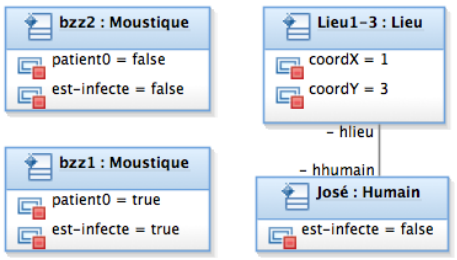

Figure 2.3. *Diagramme d'objets UML*

2.3.1.1.4. Méta-modèle

Un méta-modèle est un langage de modélisation qui permet de décrire un autre langage comme une grammaire pour une langue. En UML, le diagramme de classes représente l'ensemble des éléments permettant de décrire un modèle UML. Ceci permet au langage de se représenter ou de se définir lui-même et ainsi de définir un nouveau canevas de modélisation. Celui-ci permet d'étendre ou de spécialisé UML comme il a été proposé par les auteurs d'AML. Nous verrons cela dans la partie suivante.

2.3.1.2. *Formalisation du fonctionnement des modèles (diagrammes dynamiques)*

Cette partie présente les diagrammes analytiques ou encore appelés diagrammes dynamiques puisque permettant de décrire l'aspect dynamique du système. Elle regroupe l'ensemble des diagrammes de comportement et des diagrammes d'interaction. Sur les sept diagrammes possibles, nous présenterons ici seulement trois diagrammes : le diagramme d'activités, le diagramme d'état-transitions et le diagramme de séquences. Nous finirons en parlant des aspects vérification de cohérence. Ceci peut être réalisé sur la base des éléments de méta-modélisation et entre l'ensemble des informations portées par les différents diagrammes.

2.3.1.2.1. Diagramme d'activités

Le diagramme d'activités UML fait partie des diagrammes qui permettent de représenter le comportement d'un objet à l'aide de nœuds (d'activité, d'action, de contrôle ou d'objets) et de transitions. Les diagrammes d'activités sont adaptés pour spécifier des traitements séquentiels ou concurrentiels et permettent d'offrir une vision des flots de contrôle d'une activité à l'autre. Dans la figure 2.4, nous présentons l'activité liée au déplacement d'un moustique. Elle pourrait être vue comme l'activité associée à la méthode `go-moustique`. Ainsi, le moustique se déplace et, s'il y a un humain dans sa zone, il le pique.

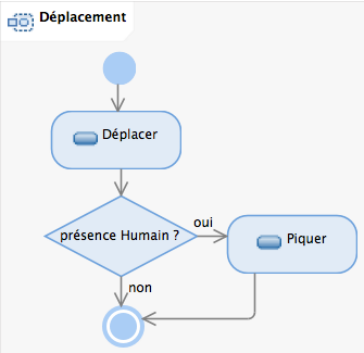

Figure 2.4. *Diagramme d'activité UML*

2.3.1.2.2. Diagramme d'états-transitions

Le diagramme d'états-transitions sert à représenter des automates à états finis (c'est à dire des entités caractérisées par un ensemble d'états et qui à chaque instant sont dans un état particulier), sous forme d'un ensemble de transitions étiquetées ou non. Un état est caractérisé par la valeur des attributs d'un système à un instant t. Une transition représente le passage instantané d'un état vers un autre et elle est généralement déclenchée par un événement. Elle peut être automatique, lorsque l'événement

qui la déclenche n'est pas spécifié. De plus, il est possible de conditionner le déclenchement d'une transition à l'aide de *gardes* : il s'agit d'expressions booléennes, exprimées en langage naturel ou en *OCL* (*Object Constraint Language*) par exemple. Dans la figure2.5, nous présentons le cycle de vie d'un moustique. Il commence sa vie en déplacement jusqu'à ce qu'il meure ou qu'il soit écrasé par un homme. Sinon il se pose pour piquer un individu et pendant l'échange de fluide peut avoir lieu la contamination de l'une ou l'autre des entités. Il est à noter que le moustique ne se repose jamais et dès qu'il est dans le même lieu qu'un humain il le pique.

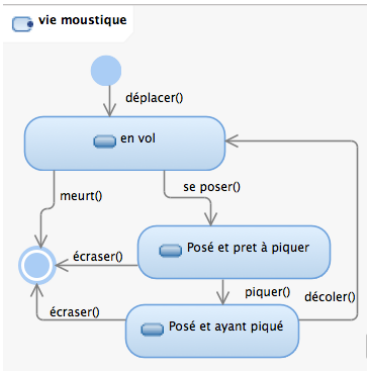

Figure 2.5. *Diagramme d'états-transitions UML*

2.3.1.2.3. Diagramme de séquences

Le diagramme de séquences permet de représenter les interactions entre des entités du système modélisé (acteurs ou objets). Il permet de voir les échanges (synchronisés ou pas) entre les entités. Il existe un langage de composition permettant d'exprimer les traitements parallèles à l'aide de cadres d'interactions qui peuvent contenir des algorithmes. Dans la figure 2.6, nous avons un échange entre les trois entités du diagramme d'objets qui montre un cycle de contamination.

2.3.1.2.4. Vérification de cohérence

Comme pour toute représentation, il est important de vérifier la cohérence du modèle. Aujourd'hui, il existe des outils permettant de faire cette vérification, tel que ATL. ATL[1] est un langage de transformation de modèle, au niveau des méta-modèles,

1. ATL pour ATLAS *Transformation Language*

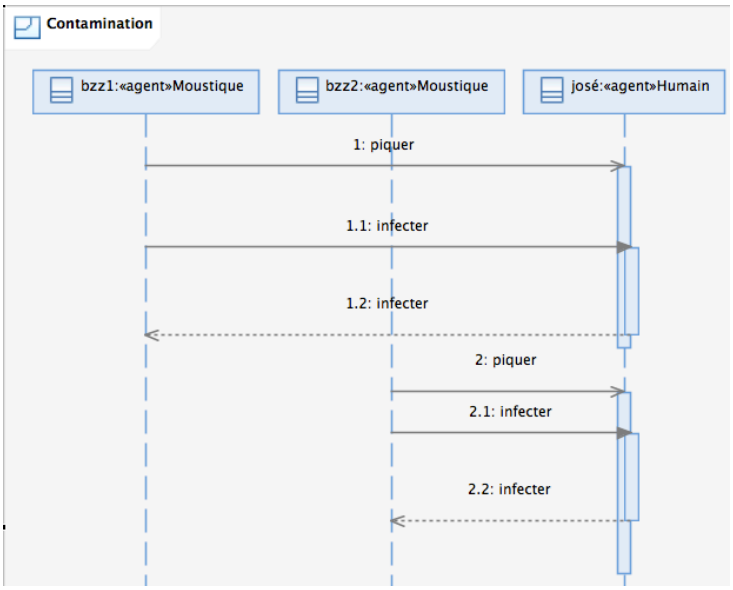

Figure 2.6. *Diagramme de séquences UML*

développé depuis 2003 au sein de l'Université de Nantes. Il a pour vocation de permettre l'expression de règles de transformation de modèle et de pouvoir les exécuter. Depuis Janvier 2007, ATL fait partie de la section M2M (*Model-To-Model*) d'Eclipse et est donc recommandé en tant qu'outil de transformation de modèle à modèle. De plus, il existe un outillage intégré comme *plugin* à la plateforme de conception Eclipse. Pour ce faire, il faut définir un méta-modèle qui permet de représenter un problème comme le montre la figure 2.7.

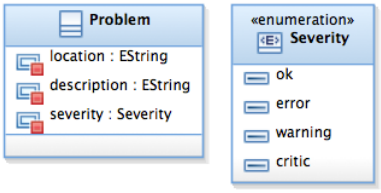

Figure 2.7. *Méta-modèle de problèmes*

Ce méta-modèle contient une méta-classe `Problem` et une méta-énumération `Severity`. Ce méta-modèle nous permet alors d'instancier des problèmes dont on peut connaître la source (attribut `location`), la description et la sévérité. La sévérité

nous permet de savoir si l'erreur est critique pour la génération de code. Dans ce cas aucun code n'est généré. En revanche, s'il existe des avertissements *warnings* n'introduisant pas d'erreurs dans le code généré alors la génération de code a bien lieu.

Si une telle modélisation peut être réalisée à l'aide d'UML standard, elle sera difficilement compréhensible au premier abord et ne pourra servir d'outil de communication.
C'est la raison pour laquelle la partie qui suit s'intéresse à l'extension AML qui a été apportée à UML.

2.3.2. *Le langage AML (*Agent Modeling Language*)*

Le langage UML étant très généraliste et orienté vers la programmation objet, de nombreux travaux ont été menés pour proposer un formalisme graphique adapté, voire dédié au paradigme agent.

Le développement, à partir du milieu des années 1990, de méthodes de conception des systèmes multi-agents (*e.g.* Gaia [ZAM 03], TROPOS [BRE 04] ou Prometheus [PAD 02]) s'est naturellement accompagné de celui de différents langages de modélisation graphiques dédiés (*e.g.* AUML [BAU 01b], AML [CER 07b]) plus ou moins basés sur le langage UML, avec pour but d'augmenter UML de concepts spécifiques au domaine des SMA. Parmi ces langages, nous allons nous intéresser plus particulièrement au langage AML (*Agent Modeling Language*) [TRE 05, CER 07b]. AML est un langage semi-formel de modélisation et de documentation d'applications à base de Systèmes Multi-Agents (SMA). Il a été développé comme une extension à UML 2.0 [OMG 03]. Les objectifs principaux des concepteurs de ce langage ont été d'inclure et de réunir les concepts des différentes architectures d'agents (notamment BDI – *Beliefs Desires Intentions*), langages et méthodes de modélisation existants. AML se veut donc adapté à tout type d'application à base d'agents et indépendant de la méthodologie et de la plateforme de développement utilisée ou du cas d'application.

2.3.2.1. *Formalisation de la structure des modèles (Agent-Groupe-Rôle)*

2.3.2.1.1. Les Agents

Cette partie met l'accent sur la description de la structure du modèle, ce qui se fait en UML principalement à l'aide de diagrammes de classes. Contrairement à l'UML qui ne permet de définir que des classes, AML affine et permet de représenter différents types d'entités intervenant dans les systèmes multi-agents et les simulations à base d'agents. La figure 2.8 est un fragment du méta-modèle complet d'AML. Elle présente la hiérarchie des entités d'AML à un niveau très général. On ne présente pas ici sa description plus détaillée. De la même manière les concepts et formalismes utilisés dans le modèle de structure sont des concepts de bas niveau. Par exemple,

nous utiliserons directement la notion d'agent sans rappeler qu'il s'agit d'une spécialisation de *AutonomousEntityType* qui hérite des entités *BehavioredSemiEntityType*, *MentalSemiEntityType*, etc.

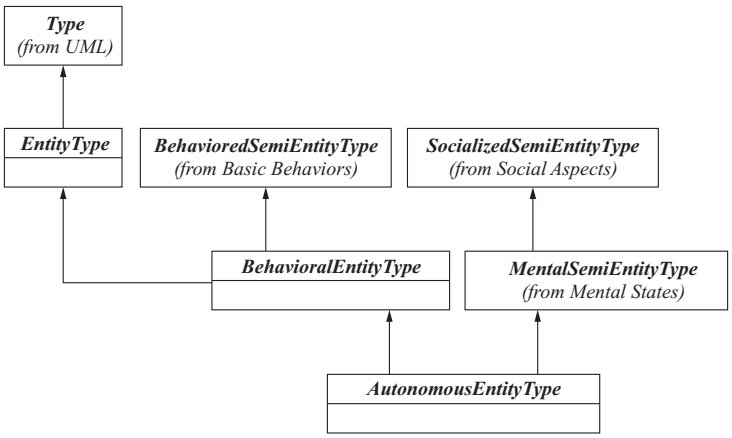

Figure 2.8. *Hiérarchie des entités dans AML*

Les entités principales sont les **agents** (entités autonomes capables de percevoir, d'interagir et d'avoir un comportement dans son environnement ; ils sont également dotés d'attitudes mentales et de capacités sociales), les **ressources** (entités sans comportements propres, mais qui sont utilisés par les agents ; leur disponibilité est donc une caractéristique importante) et les **environnements** (les entités dans lesquels évoluent les autres entités). En lien avec les capacités sociales des agents, AML permet aussi de définir des entités **rôle** et **organisation**. Ces deux notions sont proches de celles présentes dans le modèle AGR (Agent, Groupe, Rôle [FER 04]). Une organisation permet aussi de représenter un ensemble d'agents considérés à un niveau supérieur comme un agent unique.

Graphiquement chaque entité est représentée par une classe UML avec un stéréotype spécifique à l'entité et/ou une icône, qui spécifie à quel type appartient cette entité. Elle est caractérisée par une liste d'attributs, une liste d'opérations, les éléments qui le composent (le champ *part* en UML 2) et des comportements, comme illustré en figure 2.9. Les *parts* sont particulièrement utiles pour préciser les agents composant une organisation. Les comportements sont des actions complexes constituées d'opérations voire d'autres comportements. La déclinaison pour les différents types d'entités est donnée en figure 2.10.

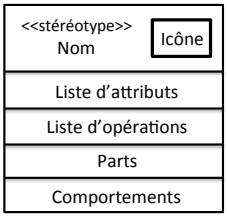

Figure 2.9. *Description d'une entité en AML*

Type d'entité	Stéréotype	Icône
Agent	<<agent>>	
Environnement	<<environment>>	
Ressource	<<resource>>	
Organisation	<<organization unit>>	
Rôle	<<entity role>>	

Figure 2.10. *Stéréotype et icône pour chaque type d'entités*

Enfin, le formalisme AML est très ouvert et il est possible de décrire de plusieurs manières différentes un même élément du système à modéliser. Aussi dans la description des éléments du formalisme qui suit, nous allons préciser les choix que nous avons fait dans notre méthodologie. Nous allons présenter la modélisation AML sur la base de l'approche AGR.

L'agent est l'élément central de la modélisation. Au niveau le plus bas, on le décrit comme dans la figure 2.9. À des niveaux plus élevés, on peut faire référence à un agent déjà décrit simplement à l'aide de son nom et de l'icône *agent*.

Un agent est identifié par un nom et pour un agent donné il pourra y avoir plusieurs instances donc plusieurs agents du même *type*. Il faut donc bien faire la distinction entre le *type* d'agent (proche du concept de classes UML) et l'agent lui-même qui sera présent dans le système (instance).

Comme le montre la figure 2.9, on trouve ensuite la liste des attributs d'un agent et la liste de ses opérations (les actions qu'il peut effectuer). La partie *parts* de l'agent restera toujours vide dans le cadre de cet ouvrage. En effet on considère qu'un agent ne peut être composé d'autres agents (comme c'est le cas dans une implémentation en

Netlogo)[2]. La dernière partie *behaviors* contient les *fragment behavior*. Ils permettent de décrire des opérations *complexes* que peut accomplir l'agent en les décomposant en plusieurs actions plus simples. On peut également utiliser des *fragment behavior* pour décrire des actions réutilisables par d'autres agents. Comme décrit dans la section 2.3.2.2.2, cette partie de l'agent ne sera remplie que si ce dernier est capable d'actions complexes. Cette notion sera décrite plus en détail dans le modèle d'interactions.

2.3.2.1.2. Les groupes

La notion de groupe n'existe pas en tant que telle en AML. On y trouve à la place des *organization unit*. On les utilise en AML pour décrire des structures organisationnelles, des environnements spécifiant des arrangements sociaux entre des entités en terme d'interaction, de rôles, de contraintes, etc. Cette approche est relativement proche de la notion de groupe telle qu'elle est introduite dans le modèle Agent/Groupe/Rôle. Aussi nous avons décidé d'utiliser l'*OrganizationUnitType* pour décrire les groupes de notre système multi-agents. De la même manière que pour les agents, il faut faire la différence entre le groupe en tant que structure et le groupe en tant qu'instance.

Dans l'approche AGR, le groupe est une notion abstraite, permettant de réunir des agents partageant des caractéristiques ou des ressources communes. En l'état, le formalisme AML, présenté dans la figure 2.9, est beaucoup trop riche. En effet, un groupe tel qu'il est défini dans le modèle AGR ne possède ni attribut ni méthode. Ce formalisme va toutefois nous permettre de modéliser la notion de groupe agentifié.

Notre modèle doit en effet pouvoir décrire le fait qu'un ensemble d'agents peut être considéré comme une entité unique à un niveau d'abstraction supérieur. Nous le représentons en AML à l'aide des *OrganizationUnitType* et d'un rôle spécifique de *leader* pour chaque groupe. Ce rôle aura toujours le nom de meneur suivi du nom du groupe auquel il se rapporte.

Dans un modèle, un groupe aura ses parties *attribute list*, *operation list* et *behavior* vides. Seule la partie *parts* sera remplie : elle permet de décrire la structure du groupe, c'est-à-dire de quoi il est composé. Un groupe peut être composé d'agents, d'autres groupes, voire des deux. On trouve également dans cette partie les différents rôles qui peuvent être joués par les agents du groupe, avec le rôle spécifique de *Leader* qui est présent dans chaque groupe.

2. Contrairement à Netlogo, d'autres plateformes de modélisation et simulation à base d'agents (comme GAMA [GRI 13b] ou Cormas [PAG 12] par exemple) permettent de définir des agents comme étant composés d'autres agents.

2.3.2.1.3. Les rôles

D'après le modèle AGR, un rôle appartient à un groupe. Dans notre modèle de structure, on le représente en plaçant tous les rôles d'un groupe dans la partie *parts* du groupe.

En AML, un rôle est décrit comme le présente la figure 2.9. La notion de rôle est telle que définie par Ferber [FER 98] pour le modèle Agent/Groupe/Rôle. Un rôle ne peut donc pas être composé d'autres rôles, pour cette raison, la partie *parts* sera donc toujours vide. Les trois autres parties peuvent être vides ou non, selon que le rôle donne accès à des attributs, des méthodes simples ou des méthodes complexes.

On notera l'existence systématique d'un rôle particulier de *Leader* pour chaque groupe agentifié. Pour ce rôle, la partie *attribute list* (respectivement *operation list*) permet de décrire la liste des attributs (respectivement méthodes) du groupe agentifié, c'est-à-dire de l'agent jouant le rôle de leader du groupe. La partie *behaviors* permet de décrire les actions complexes qu'est capable de réaliser le groupe agentifié (et donc l'agent qui joue le rôle de *Leader*). Comme pour un rôle classique, ces parties peuvent être vides.

D'après la définition d'un rôle dans le modèle AGR, il nous faut encore modéliser deux choses : le fait qu'un rôle puisse nécessiter des prérequis pour être joué et le fait que certains rôles puissent permettre de diriger d'autres agents. AML étant basé sur UML 2 et compatible avec celui-ci, les prérequis peuvent être décrits à l'aide de contraintes OCL sur la relation de *"jouer un rôle"* dont la représentation graphique est présentée dans la figure 2.12. Pour ce qui est de la hiérarchie entre les rôles, elle est exprimée à l'aide des liens entre les rôles comme le présente la figure 2.11.

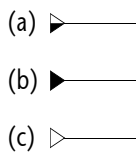

Figure 2.11. *Les types de lien entre les rôles en AML*

Il existe deux types de relations entre les entités : les relations "pair-à-pair" et les relations "maître/esclave". Pour représenter ces types de relations, il existe trois types de lien entre deux rôles. Les liens de "pair-à-pair" (a) sont représentés par des triangles blancs et noirs. Les relations "pair-à-pair" sont des relations entre des entités de même statut social et d'autorité égale.

Les relations de type "maître/esclave" sont représentées par les liens (b) et (c). Le lien (b), un triangle noir, représente le dirigeant (ou *superOrdored* selon AML). Un dirigeant peut contraindre le comportement des agents qu'il commande. De l'autre côté de la relation on trouvera un lien (c) formé d'un triangle blanc, représentant le subordonné. Un lien de type (a) est toujours associé avec un autre lien (a) et un lien de type (b) est toujours associé avec un lien de type (c).

Maintenant que nous pouvons relier des rôles entre eux, voyons comment relier des rôles à des agents.

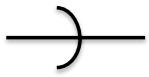

Figure 2.12. *Jouer un rôle en AML*

La figure 2.12 représente l'association entre un rôle et un agent qui signifie que l'agent joue le rôle en question. Le demi-cercle est orienté du côté du rôle. On peut également préciser la multiplicité du rôle, c'est-à-dire le nombre maximum d'agents pouvant jouer ce rôle dans ce groupe au même moment.
Nous avons donc présenté les trois composantes essentielles du modèle AGR, à savoir les agents, les groupes et les rôles. Il existe toutefois encore deux entités utilisées dans le modèle de structure : il s'agit de l'environnement et des ressources.

2.3.2.1.4. L'environnement et les ressources

L'environnement est l'entité logique et physique qui entoure toutes les entités du système. Sa représentation AML est présentée dans la figure 2.9. Il ne peut y avoir au maximum qu'un seul et unique environnement dans le modèle de structure. Il est également possible que l'environnement ne soit pas présent s'il n'apporte rien à la modélisation.

L'environnement étant l'entité globale de notre système, nous avons considéré que les ressources lui étaient rattachées. Ainsi, la partie *Parts*, renommée *Resource List* contient la ou les ressources du système. Les trois autres parties de l'environnement peuvent être utilisées. On ne considère de l'environnement que les variables globales du système (comme par exemple la température) et les opérations qui permettent de les modifier.

Les ressources sont des entités physiques (par exemple des matières premières dans un système de production) ou des entités informationnelles (par exemple une base de données). On considère dans notre modèle de structure des ressources *simples* c'est-à-dire non composées d'autres éléments. Pour cette raison, une ressource n'aura

jamais de *Parts* dans sa représentation (voir la figure 2.9). Une ressource pourra posséder un ou plusieurs attributs (comme sa capacité par exemple). Enfin, dans le cas d'une ressource informationnelle, on considère qu'elle peut contenir des opérations simples ou complexes. Dans le cas d'une base de connaissances par exemple, ces opérations pourront être utilisées par des agents y ayant accès.

2.3.2.1.5. Représentation de l'exemple fil rouge avec AML

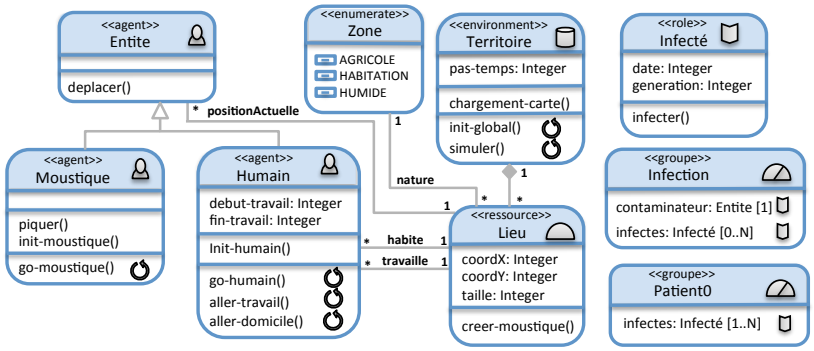

Figure 2.13. *Représentation AGR avec AML de l'exemple fil rouge*

Dans la représentation AML de l'exemple fil rouge, nous avons volontairement pris le contre-pied de la représentation UML pour formaliser le lien des infections. En effet, puisque AML nous permet d'adopter l'approche Agent-Groupe-Rôle nous avons décidé de l'utiliser pour représenter l'infection. Ici, nous avons deux rôles. Le premier rôle concerne le `contaminateur`. Rôle particulier dans un groupe, il permet de connaître la source qui a contaminé l'ensemble des entités du groupe `Infection`. Le second rôle concerne l'`infecté`. Il indique, lorsqu'une entité est infectée, la date de son infection et la génération. Il permet aussi d'acquérir la capacité d'infecter (opération `infecter()`) et de devenir à son tour un contaminateur d'un groupe `Infection`. Le second groupe `patient0` permet de gérer les moustiques infectés à l'état initial du système. Pour le reste, on retrouve les mêmes éléments que pour le modèle UML. Une petite variation concerne la mise en exergue (à l'aide de la flèche circulaire) des comportements complexes qui sont basés sur d'autres opérations ou comportement du systèmes comme par exemple `aller-travail` qui utilise `déplacer` pour aller de `Lieu` spécifique qui sont la maison au travail.

Ceci termine la partie descriptive, intéressons-nous maintenant à la partie analytique. Elle contient les interactions et les actions.

2.3.2.2. *Formalisation du fonctionnement des modèles (interactions et actions)*

Dans cette partie, nous allons présenter les éléments de modélisation permettant de décrire la partie dynamique des modèles. Ceci concerne plus particulièrement les interactions et les actions.

2.3.2.2.1. Modèle d'interactions

On distingue deux types d'interaction : les interactions d'un agent avec un ou plusieurs autres agents, et les interactions entre un agent et l'environnement, support des agents et des ressources. Ici, nous considérons plusieurs interactions qui peuvent être regroupées en cinq grands types :

– **la communication :** on peut citer trois types de communication, l'envoi de messages, le tableau noir et la communication par marques (par exemple, les phéromones). Un message peut avoir plusieurs conséquences sur un agent : changer son état, lui faire envoyer une réponse, lui faire effectuer une action particulière ;

– **la réaction :** on distingue les actions des agents et les effets que ces actions produisent. Une action effectuée peut avoir des conséquences sur un ou plusieurs autres agents, ainsi que sur l'environnement ;

– **les actions coopératives et/ou complexes :** certaines actions ne peuvent être effectuées par un seul agent et demandent une coopération entre plusieurs d'entre eux. D'un autre côté, certaines actions complexes peuvent être divisées en plusieurs actions plus simples. C'est particulièrement le cas avec les actions dites *de haut niveau d'abstraction* dont nous parlerons plus en détail plus tard ;

– **l'ordonnancement :** grâce au modèle d'actions nous fournissant les dates de début et de fin de chacune des actions effectuées, nous pouvons vérifier si un ordonnancement est possible ou non. De plus, nous pouvons produire un ordonnancement naïf le cas échéant. Ainsi il est possible de vérifier qu'un agent n'accomplit pas une action avant qu'une précédente ne soit terminée (par lui ou par un autre agent) ;

– **la connaissance et l'apprentissage :** les agents sont capables de mémoriser, d'apprendre et de modifier leur base de connaissances pendant l'évolution du système. Cette partie étant à elle seule un sujet très important, ces deux notions sont très restreintes dans notre modèle et leur approfondissement fait partie des évolutions possibles.

2.3.2.2.2. Les actions

Le formalisme AML étant ouvert quant aux choix de modélisation, un même modèle, même très simple, peut être décrit de plusieurs manières différentes avec ce formalisme. Nous allons donc préciser et spécifier certains choix de modélisation.

Il est possible de décrire les actions et capacités des agents à différents niveaux. On fait donc le choix de modélisation suivant le type d'action. L'élément discriminant est le fait de n'avoir ni précondition ni postcondition *spéciales*. Nous considérons une

condition comme étant *spéciale* si elle ne porte pas seulement sur la capacité et les connaissances de l'agent quant à l'accomplissement de l'action.

Ainsi une **action simple** est sans précondition ni postcondition *spéciales*. Elle sera décrite directement dans la partie organisationnelle de l'agent.

2.3.2.2.3. Les actions complexes

Les **actions coopératives** et les **actions complexes** sont décrites dans le modèle de structure à l'aide du concept d'AML *fragmentBehaviour*.

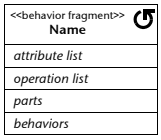

Figure 2.14. *Description d'un comportement en AML*

La figure 2.14 présente ce qu'en AML on traduit par *fragment de comportement*. Ils sont utilisés dans deux cas: décrire des comportements qui pourront être réutilisés ou décomposer un comportement complexe en plusieurs plus simples. Dans le premier cas, cela signifie qu'on fait correspondre à un fragment de comportement plusieurs opérations. Ainsi tout agent possédant ce fragment de comportement possède ces opérations. Nous ne les utilisons pas de la sorte, mais pour décrire des actions dites *complexes*. Aussi, les parties *attribute list* et *parts* restent toujours vides. Si l'action complexe se divise en plusieurs actions simples, celles-ci apparaîtront dans la partie *operation list*. Si l'action complexe se divise en plusieurs actions elles aussi complexes, elles apparaîtront dans la partie *behaviors*.

2.3.3. *UML versus AML*

UML est très largement répandu. Son expressivité en fait un outil de modélisation et de communication adapté à nos besoins. On peut l'utiliser en complément d'autres langages et/ou formalismes, voire lui apporter des extensions. Le langage OCL permet également d'exprimer des propriétés du système.

L'approche UML dans une mise en œuvre de système multi-agent peut engendrer une complexité supplémentaire. Notamment, lorsque le modèle doit intégrer des paradigmes SMA qu'il ne permet pas de proposer nativement (comme par exemple l'AGR). Il est alors nécessaire de les redéfinir ou d'ajouter des informations pour faire

le lien avec les implémentations. Le risque engendré est la perte des aspects illustratifs et intuitifs. Ainsi, la complexité ne serait pas due au système étudié mais aux éléments de modélisation.

Par exemple, comment exprimer les différents niveaux d'abstraction qui composent un système ? Comme modéliser l'autonomie des entités à chacun de ces niveaux, avec une perception, des connaissances, des capacités et des accès aux ressources différents ?

Il est clair que le formalisme AML qui a été défini pour cela propose ces concepts. Ses détracteurs diront qu'il est hélas peu répandu. Ceci vient en partie du fait qu'il n'existe que peu d'outils supportant ce langage.

Il est important de noter que le langage AML est extrêmement puissant et modulaire ce qu'il doit en partie à son héritage d'UML. En contrepartie, il n'est pas aisé à saisir dans son ensemble. De plus toute sa complexité n'est pas forcément utile dans le cadre de la simulation à base d'agents. Dans cet ouvrage, nous n'avons pas présenté la liste exhaustive des concepts existants dans AML, nous nous sommes limités à ceux qui nous paraissent pertinents pour le domaine de la simulation à base d'agents et qui ont été utilisés dans l'exemple. Pour avoir une vue complète, le lecteur peut se référer à l'ouvrage de référence sur AML [CER 07a].

Dans le cadre de cet ouvrage, nous nous intéressons au lien entre les modèles en particulier UML et l'outil Netlogo.

2.3.4. *Autres variations d'UML*

S'il existe des solutions de modélisation pour les systèmes multi-agents, elles sont pour la plupart dédiées à un domaine particulier et ne peuvent être utilisées dans le cas général. On pourra toutefois noter plusieurs travaux proposant un langage de modélisation et de spécification : FIPA ACL [INT 97], KQML [FIN 94], TAO [SIL 03], OPM/MAS [STU 03], AUML [ODE 00, BAU 01a] et [ODE 99]...

Le plus abouti est l'approche AUML. Agent UML [ODE 00, BAU 01a] propose des mécanismes pour la modélisation de protocoles d'interaction dans les systèmes multi-agents. Pour ce faire, ils introduisent une nouvelle classe de diagramme dans UML : les diagrammes de protocole. Ils étendent les diagrammes d'état et de séquences d'UML de plusieurs manières. Ces extensions particulières introduisent les rôles des agents, leurs séquences d'exécutions simultanées, étendent la sémantique des messages et modélisent les protocoles d'interaction.

Agent UML a été proposé et accepté pour son inclusion dans la norme FIPA'99. Toutefois, AUML ne propose pas de solution graphique au problème des groupes agentifiés.

2.3.5. *Formaliser des changements de comportement en UML*

Le concept de rôle que nous avons vu dans l'approche AGR est également utile pour spécifier un comportement qui va changer dans le temps au niveau d'une même entité. Si ce concept est directement disponible dans AML, ce n'est pas le cas lorsque le modélisateur prend le parti d'utiliser une formalisation UML. Dans cette section nous allons montrer comment le modélisateur peut structurer son modèle pour représenter des changements de comportement, et ce en se servant d'un patron de conception (*Design Pattern*) comme l'Acteur-Rôle [COA 97].

Le patron de conception Acteur-Rôle consiste à représenter les attributs et les actions liés à un comportement dans une classe spécifique (la classe rôle) associée à la classe Agent. Comme l'explique Bommel, *« lorsque l'on s'intéresse à la représentation d'humains ou d'animaux qui peuvent évoluer, se transformer et changer de comportement au cours de leur vie, il est utile [..] d'associer un comportement spécifique à l'agent jouant ce rôle, [..] pour lui permettre de changer de rôle (et donc de comportement) au cours du temps »* [BOM 09]. Ainsi, lorsqu'une entité change de comportement, il n'y a pas de modification en profondeur à faire au niveau de cette entité, il suffit de lui associer un nouveau comportement. Appliqué à des entités spatiales, nous parlerons plus volontiers d'état plutôt que de rôle. L'application de ce patron consiste alors à spécifier dans une classe distincte de l'entité spatiale, les actions et les attributs liés à un certain état. Cette structure va s'avérer particulièrement pertinente pour modéliser des occupations de l'espace dont les états sont porteurs de dynamiques propres. Elle permet également d'organiser de manière claire ces dynamiques et donc de mieux communiquer sur le contenu du modèle.

Nous proposons d'étendre l'exemple Fil rouge pour présenter l'application du patron Acteur-Rôle à des dynamiques d'entités spatiales. Nous illustrerons donc son utilisation à partir d'un cas bien connu dans l'étude des changements d'occupation du sol.

Dans l'exemple de la figure 2.15, nous retrouvons le patron Agent-Rôle au niveau de l'association entre la parcelle et son occupation du sol. Dans ce modèle une instance de parcelle ne change pas au cours du temps, sa surface demeure constante. Par contre, son occupation du sol est susceptible de changer. Plusieurs types d'occupation du sol sont possibles, ayant chacune leur propre dynamique. Une occupation du sol de type culture (spécialisation) peut calculer sa production une fois que son âge a atteint le nombre de jours avant récolte. De son coté, une forêt a comme action dans ce modèle de stocker du carbone. Grâce à cette structure, l'opération d'un agent tiers qui consisterait à mettre en culture une parcelle préalablement en forêt, revient à remplacer l'association de la parcelle à une instance de la classe forêt par une association à une instance de la classe Colza par exemple.

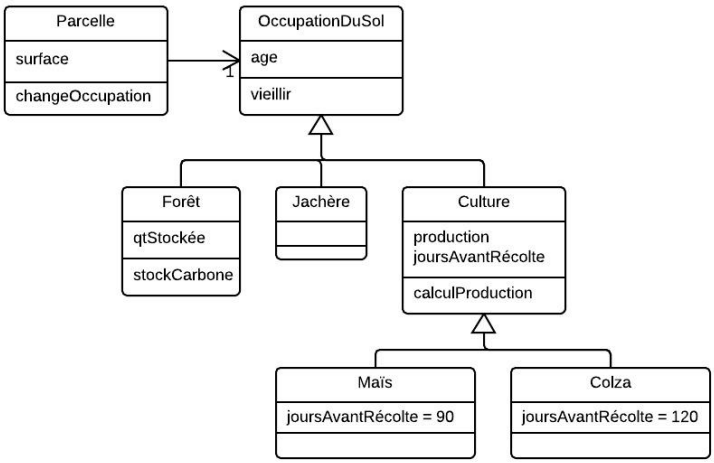

Figure 2.15. *Un diagramme de classes pour modéliser des changements d'occupation du sol (figure adaptée de [LEP 05])*

Si cette architecture a le grand avantage de sa clarté, son implémentation dans NetLogo peut s'avérer difficile du fait qu'un seul niveau d'héritage est possible dans la plateforme et qu'il est réservé à la classe *turtle*. Dans les faits, on se retrouvera bien souvent amené à spécifier les attributs et à coder les opérations de tous les types d'occupation (ou rôles) dans une même classe, et il faudra alors ajouter des conditions pour s'assurer que l'entité ne puisse pas exécuter des méthodes correspondant à un autre type que celui possédé à l'instant courant. De même, il faudra veiller à réinitialiser correctement les attributs d'un type à chaque changement d'occupation du sol.

Tout cela permet d'illustrer la différence entre un modèle, aussi bien conçu soit-il, permettant de prendre en compte l'ensemble des paradigmes et la réalité de l'implémentation. En effet, celle-ci devra être faite dans un environnement ou avec des langages qui nécessitent au mieux une certaine vigilance dans la réalisation voire des adaptations profondes des concepts comme par exemple l'absence de notion d'objets.

2.4. Description et documentation des modèles agents

Cette partie aborde la question de la description et de la documentation d'un modèle multi-agent. La première partie est consacrée au rappel des rôles de la documentation d'un modèle et de sa standardisation : aide à la compréhension et à l'explicitation de la structure et du fonctionnement du modèle, à la diffusion et à la communication, à la comparaison, à la réplication, à l'appropriation et à la réutilisation, à la complétude, etc. La seconde partie présente en détail un protocole devenu un standard de facto

pour la description de modèles multi-agents : le protocole ODD [GRI 06, GRI 10]). Un bref rappel de l'historique de ce protocole est donné en guise d'introduction en précisant les objectifs de sa création. Les différents éléments qui définissent le protocole ODD (version révisée) sont ensuite abordés : (1) objectif du modèle, (2) entités, variables d'état et échelles, (3) fonctionnement général du modèle et ordonnancement des processus, (4) concepts qui sous-tendent la conception, (5) initialisation, (6) données d'entrée, (7) sous-modèles. La troisième partie illustre à travers différents exemples la mise en œuvre d'ODD en pointant des erreurs d'utilisation ou d'interprétation fréquemment commises. Cette section montre également le lien existant entre les diagrammes élaborés durant la phase de conception du modèle et les composants du protocole.

2.4.1. *Décrire et documenter : pourquoi ?*

Les définitions que donnent Jean-Louis Le Moigne [LEM 90] des modèles et de la modélisation mettent en exergue l'intelligibilité comme une des propriétés capitales d'un modèle :

– un modèle est une représentation intelligible, artificielle, symbolique, de situations dans lesquelles nous intervenons,

– la modélisation est l'action d'élaboration et de construction intentionnelle de modèles susceptibles de rendre intelligible un phénomène perçu complexe, et d'amplifier le raisonnement de l'acteur projetant une intervention délibérée au sein du phénomène, ce raisonnement visant notamment à anticiper les conséquences de ses projets d'actions possibles.

Un modèle constitue ainsi un ensemble organisé de connaissances représentant un système complexe sur lequel on se pose des questions, dont on souhaite comprendre son fonctionnement actuel et, le cas échéant, son évolution passée, mais aussi imaginer son évolution potentielle selon divers scénarios, et souvent agir dessus pour orienter son évolution vers des situations souhaitées.

La documentation d'un modèle constitue ainsi un enjeu fondamental pour la mobilisation, la compréhension, et le partage de la connaissance portée par celui-ci, et son utilisation de la manière la plus efficiente possible. Elle doit permettre de le décrire de manière pédagogique, explicite, non ambiguë, et pertinente pour un large public.

2.4.1.1. *Décrire de manière pédagogique la connaissance portée par le modèle*

Le caractère pédagogique d'une documentation est de permettre d'aborder de manière progressive les différents niveaux de description du modèle, du plus global au plus détaillé, car on ne peut rentrer directement dans le détail ou le code d'un modèle complexe : il est indispensable d'avoir d'abord une vue d'ensemble, d'en comprendre

le contexte et les objectifs, de comprendre ce que fait le modèle avant de comprendre comment il est fait, puis de se focaliser peu à peu sur ses composants globaux, leurs interactions, et d'aboutir progressivement au détail des données manipulées et des processus modélisés. C'est précisément dans cet objectif qu'a été développé le protocole ODD (*Overview, Design concepts, Details*) qui va être décrit dans la section suivante.

2.4.1.2. *Expliciter sans ambiguïté la structure et le fonctionnement du modèle*

Aucun des aspects d'un modèle considéré comme stabilisé ne doit demeurer implicite ou flou, au point de manquer une brique de savoir pour comprendre le modèle ou le rendre opérationnel. C'est une difficulté qui apparaît souvent lorsqu'on affine un modèle conceptuel pour le transformer en un modèle informatique opérationnel. Cette étape fait généralement apparaître des manques, des ambiguïtés ou des incohérences qui nécessitent de revenir sur le modèle conceptuel pour en préciser ou redéfinir certains aspects. Il y a une sorte de dialectique entre modèle conceptuel et modèle informatique qui nécessite plusieurs cycles pour éliciter progressivement la connaissance portée par les modèles. La plus grande difficulté réside sans doute dans la manière dont la documentation du modèle rend compte de l'organisation du système modélisé, des interactions entre ses composants, et des propriétés émergentes d'un niveau d'organisation par rapport au(x) niveau(x) sous-jacent(s), tout en gérant un historique et un justificatif des décisions prises pour aboutir à une version stabilisée du modèle.

Dans ce processus de modélisation, la notion d'ambiguïté revêt un caractère particulier : en début de conception d'un modèle, accepter une part d'ambiguïté peut se révéler utile pour ne pas focaliser ou bloquer la réflexion sur un point susceptible de poser problème entre partenaires participant à la modélisation ; mais il importera de la lever plus tard dans l'affinage de la conception, en précisant les conditions conduisant à l'accepter ou la refuser. Une ambiguïté temporaire peut ainsi apporter une souplesse utile au processus de modélisation, notamment lorsqu'il est mené de manière collective.

2.4.1.3. *Prendre en compte la multiplicité des usagers du modèle*

La documentation d'un modèle se doit d'être pertinente vis-à-vis de ses multiples usagers aux différents stades de son élaboration et de sa mise en œuvre. Ainsi, elle doit lui permettre d'être compris par les différentes parties prenantes au processus de modélisation, avec des personnes ayant leurs propres savoirs (scientifique, institutionnel, technique, empirique) et centres d'intérêt disciplinaires (thématiques, informatique). Un modèle doit être compris par les différents types d'usagers qui ont contribué à son développement, mais aussi par d'autres personnes que les parties prenantes initiales, que ce soit pour l'utiliser ou pour le faire évoluer. Cela implique plusieurs niveaux, non forcément hiérarchiques, de documentation. L'utilisation de formes de documentation diverses (écrit, graphique, diagramme, vidéo...) peut également aider à ce qu'un modèle soit compris et compréhensible par différents types de public. De même que

la construction du modèle d'un système complexe nécessite la prise en compte de plusieurs points de vue [LEG 97], de même la documentation d'un modèle doit permettre d'appréhender et de comprendre ces divers points de vue.

2.4.1.4. *Inclure la méta-connaissance mobilisée pour construire le modèle*

La documentation doit également permettre de justifier pourquoi tel aspect a été retenu dans le modèle, et tel autre a été rejeté. V. Grimm [GRI 99] notait que l'un des avantages des modèles individu-centrés était de pouvoir intégrer des savoirs empiriques, avec toute la difficulté de définir les critères permettant de décider si une information empirique doit ou non alimenter le modèle. Une documentation pertinente se doit de préciser ces critères.

Ainsi, documenter un modèle, c'est non seulement décrire la connaissance qu'il contient, mais aussi décrire la méta-connaissance qui a permis d'éliciter cette connaissance (*i.e.* les hypothèses d'un modèle).

2.4.1.5. *Permettre la réplication du modèle*

La documentation associée à un modèle devrait permettre sa réplication. À partir de l'analyse de quelques exemples de réplication, P. Bommel [BOM 09] a mis en exergue l'insuffisance des descriptions de nombreux modèles, avec plus particulièrement des faiblesses sur les spécifications relatives à la gestion du temps et des interactions. Il considère nécessaire cette étape de réplication, à faire exécuter autant que possible par d'autres modélisateurs que les concepteurs initiaux, pour renforcer la fiabilité des simulateurs et permettre une véritable réfutation des modèles. La littérature ne s'est saisie que très récemment de cette question à propos des modèles multi-agents [GRI 06], notamment parce que l'usage s'est considérablement développé dans de multiples disciplines traitant de systèmes complexes. En proposant un protocole comme ODD, cela a permis une première étape pour rendre l'écriture et la lecture de modèles multi-agents plus faciles et plus efficaces, et aboutir à une description suffisamment complète de tels systèmes pour permettre leur réplication. Une première révision de ce protocole a été publiée en 2010 [GRI 10] sur la base des retours de nombreux chercheurs ayant commencé à l'utiliser.

2.4.1.6. *Accompagner le cycle de développement du modèle*

La notion d'écriture et de lecture d'un modèle renvoie sur deux aspects de son cycle de vie : d'une part, comment faciliter la création d'un modèle (écriture efficace), d'autre part, comment en faciliter l'utilisation et son intégration à d'autres modèles (lecture efficace), ces 2 aspects n'étant pas séparés dans le temps mais intimement liés. En effet, la phase d'écriture suppose l'intervention d'un certain nombre de personnes qui doivent pouvoir partager l'état d'avancement du modèle au fur et à mesure de son développement et donc être capable de le lire de la manière la plus efficace possible. Symétriquement, une fois un modèle considéré dans un certain stade d'achèvement,

pouvoir le lire conditionne l'écriture de nouvelles évolutions de celui-ci, ou son intégration dans des modèles plus englobants.

Les modèles complexes sont bien souvent décomposables en un ensemble de sous-modèles en interaction mutuelle, chacun d'entre eux constituant un modèle à part entière, de nature éventuellement différente des autres. L'intégration de plusieurs modèles nécessite donc une bonne description de chacun d'eux, avec notamment la définition d'interfaces de communication d'informations entre eux les plus efficaces possibles.

2.4.1.7. *Développer des formes pertinentes de visualisation d'un modèle*

Comment développer des formes de visualisation les plus pertinentes possibles pour expliciter les processus constitutifs du modèle, et pour présenter, analyser et interpréter les sorties des simulations de celui-ci ? La visualisation n'intervient pas uniquement pour rendre opérationnel un modèle conceptuel, elle permet aussi d'éliciter les connaissances pour construire le modèle conceptuel [BEC 03]. Ainsi, développer une visualisation graphique d'une loi mathématique associée à un sous-modèle permet d'expliciter et visualiser les effets de cette loi sous diverses hypothèses et, partant, de mieux la calibrer, voire en changer la structure. Ainsi on peut donner à comprendre en permettant de justifier les résultats, de remonter la chaîne des causalités et interactions ayant permis de les obtenir, d'appréhender leur raison d'être, et donc de mieux comprendre le système étudié. Les interfaces graphiques favorisent la communication entre partenaires, et la médiation éventuelle en cas de difficulté d'obtention d'une vision partagée. Elles concourent à la transparence des modèles, c'est-à-dire la possibilité d'être compris assez facilement par un large éventail de personnes [WAL 77].

L'analyse et la compréhension du comportement des modèles est un des facteurs-clés de leur validation, le mot validation étant pris ici au sens de «être utilisables dans un but donné selon des critères spécifiés» [RYK 96] et "être bien fondés et justifiables" [SIN 00].

Visualiser un modèle, c'est visualiser la connaissance qu'il contient, en particulier celle sur sa dynamique et son organisation pour lesquelles la documentation statique ne suffit pas. En d'autres termes, la visualisation d'un modèle peut être considérée comme une forme dynamique de documentation. UML et AML que nous avons précédemment abordés en sont de très bons exemples.

2.4.2. *Décrire et documenter : comment ?*

Face à la difficulté récurrente d'obtenir une documentation complète et homogène des modèles multi-agents, un protocole a récemment été proposé pour décrire ces modèles de manière plus formelle. Il s'agit du protocole ODD (*Overview, Design*

concepts, and Details) proposé par Grimm *et al.* [GRI 06]. Ce protocole propose une structure prédéfinie de documentation permettant de préciser l'objectif du modèle, ses constituants, et la manière dont les propriétés spécifiques aux systèmes multi-agents sont prises en compte (émergence, adaptation, etc.). La première version du protocole ODD a été publiée en 2006 [GRI 06]. Après une mise à l'épreuve au sein de la communauté scientifique, une révision a été proposée en 2010 [GRI 10, POL 10, RAI 11]). Aujourd'hui, ODD est devenu un standard de facto pour décrire et communiquer un modèle agent. Il a déjà été adopté par de nombreux modélisateurs (par exemple [POL 08, NAI 10, CAI 13]).

2.4.2.1. *Les composants du protocole ODD*

ODD propose de décrire un modèle multi-agent en distinguant trois grandes parties (tableau 2.1) : les éléments qui fournissent une vue générale du modèle (*Overview*), les éléments de conception du modèle (*Design concepts*), et les éléments qui détaillent le fonctionnement du modèle (*Details*). Nous présentons chacun de ces éléments ci-dessous dans l'ordre préconisé par le protocole.

2.4.2.2. *Vue d'ensemble (Overview)*

La première partie d'ODD a pour but de fournir une vision synoptique du système modélisé, tant du point de vue de sa structure que de sa dynamique. Elle est composée de trois éléments principaux.

2.4.2.2.1. Objectif (*Purpose*).

Il s'agit d'indiquer de manière concise mais précise l'objectif du modèle. À quoi sert-il ? Quelle est la question de modélisation abordée ? Ce composant du protocole définit le « Quoi ? ».

2.4.2.2.2. Entités, variables d'états, échelle (*Entities, state variables, and scale*).

Une fois l'intérêt du modèle annoncé, on précise ses différents constituants. Quelles sont les entités représentées ? Quels sont les attributs qui les caractérisent ? Quelles sont les échelles spatiales et temporelles retenues ? En d'autres termes, « De quoi » est fait le modèle ?

Les entités concernent toutes les catégories du modèle (agents et groupes d'agents, entités spatiales, environnement global). Elles sont décrites par des propriétés, qualifiées de variables d'états dans ODD, dont les valeurs évoluent ou non au cours de la simulation (exemples : nom d'un agent, stratégie de comportement d'un groupe d'agents, modalité d'occupation du sol d'un patch, taux global de prélèvement d'une ressource). Les valeurs initiales sont fournies à l'initialisation du modèle. Ces variables d'états (quantitatives, qualitatives nominales ou ordinales) ne doivent pas inclure les variables dérivées ou agrégées, dont les valeurs résultent d'une combinaison ou

d'un calcul effectué à partir d'autres variables (exemples : quantité totale de nourriture consommée au cours de la simulation, densité moyenne d'animaux à l'hectare, distance au plus proche voisin).

Concernant le choix des échelles spatiales et temporelles, il s'agit d'une part de préciser la résolution spatiale des patches ou la superficie que couvre le patch dans la réalité et d'autre part, la résolution temporelle ou la durée réelle que représente un pas de temps dans le modèle. L'horizon temporel est également spécifié (durée ou longueur prévue de la simulation).

ODD (Version originale)		**ODD (Traduction française)**	
Overview	1. Purpose	Vue d'ensemble	1. Objectif
	2. Entities, state variables, and scale		2. Entités, variables d'état, échelle
	3. Process overview and scheduling		3. Processus et ordonnancement
Design concepts	4. Design concepts	Conception	4. Eléments de conception
	- Basic principles		- Principes
	- Emergence		- Émergence
	- Adaptation		- Adaptation
	- Objectives		- Objectifs
	- Learning		- Apprentissage
	- Prediction		- Prédiction
	- Sensing		- Perception
	- Interaction		- Interaction
	- Stochasticity		- Stochasticité
	- Collectives		- Coopération / Agrégation
	- Observation		- Observation
Détails	5. Initialization	Détails	5. Initialisation
	6. Input data		6. Données d'entrée
	Sub-models		7. Sous-modèles

Tableau 2.1. *Les composants du protocole ODD groupés en trois catégories (d'après Grimm et al. [GRI 06, GRI 10])*

2.4.2.2.3. Processus et ordonnancement (*Process overview and scheduling*)

Après la structure, c'est la dynamique du modèle qui est décrite. Comment se comportent les entités à chaque pas de temps ? Comment évolue l'environnement ? Dans quel ordre est organisé la simulation ? L'ensemble des processus et la manière dont ceux-ci sont enchaînés est ainsi mentionné.

Les processus sont désignés par des verbes (exemples : se déplacer, fuir, mettre à jour la quantité d'herbe consommée) et font référence aux sous-modèles détaillés en troisième partie du protocole. Le fonctionnement proprement dit des processus n'est donc pas renseigné à ce niveau. Seuls la liste des actions effectuées et leur ordre d'exécution sont fournis. Les actions concernent celles réalisées par les entités du modèle (les agents, les patches) mais aussi celles exécutées par les entités de plus haut niveau comme le modèle lui-même ou l'*observer* (pour la mise à jour d'indicateurs globaux et des graphiques associés).

Le protocole ODD n'impose pas un formalisme particulier pour présenter ces différentes informations. Toutefois, il est suggéré d'utiliser un diagramme de classes UML pour décrire la structure du modèle (voir section 2.3.1.1). Il est également possible de décrire l'ordonnancement des processus à l'aide d'un diagramme d'activités UML (voir section 2.3.1.2).

2.4.2.3. *Éléments de conception (Design concepts)*

Informer sur les propriétés émergentes du modèle, les hypothèses émises et les capacités dont certains agents sont dotés (adaptation, apprentissage) constituent des éléments de documentation importants pour l'interprétation des résultats et ces éléments sont souvent peu formalisés. Dans cette partie, ODD propose une liste de onze items qui renseignent non pas sur le fonctionnement du modèle mais sur la manière dont les concepts qui font la particularité de la modélisation agent ont été pris en compte. L'ensemble de ces concepts peut être explicité en répondant aux différentes questions énoncées par le protocole sous forme de checklist [GRI 10].

Le protocole ODD permet de ne pas renseigner l'ensemble des concepts proposés dans cette partie car selon le modèle développé, certains de ces concepts peuvent être inappropriés. Par ailleurs, il est envisageable de rajouter certains concepts qui sont propres à l'utilisateur mais dans ce cas, il convient de préciser explicitement qu'il ne s'agit pas d'éléments prévus par le standard.

2.4.2.4. *Détails (Details)*

Après avoir dressé l'esquisse du modèle avec ses principales dynamiques, cette troisième partie aborde les détails techniques qui doivent permettre de ré-implémenter le modèle dans son ensemble. ODD propose de renseigner à ce niveau trois éléments : l'initialisation, les données d'entrée et les sous-modèles.

2.4.2.4.1. Initialisation (*Initalization*)

La documentation doit spécifier toutes les conditions initiales de la simulation. Combien y a-t-il d'agents ? Comment sont-il distribués dans l'espace ? Quelles sont les valeurs initiales des variables et paramètres ? Quel est l'état de l'environnement

Principes	Quels sont les concepts, les hypothèses ou les théories sous-jacentes à la conception du modèle ? A quel niveau sont-ils intégrés dans le modèle ?
Émergence	Quels sont les phénomènes émergents non prévisibles dans le modèle qui résultent des interactions ou de l'adaptation des agents ? Quels sont les phénomènes émergents attendus qui découlent des règles introduites dans le modèle ?
Adaptation	Les agent gardent-ils toujours le même comportement ? Ont-ils une capacité d'adaptation au cours de la simulation ? Ont-ils le choix de se comporter selon plusieurs alternatives ? Quelles sont les règles de décision dictant ce choix ? Quelles sont les conditions d'un changement de comportement éventuel ?
Objectifs	Les agents cherchent-ils à atteindre explicitement ou implicitement un but en rapport avec leur comportement adaptatif ? Quel est ce but ? Est-il mesuré ? Avec quel indicateur ou quel critère ? Quelle est la fonction d'utilité ou *fitness* ?
Apprentissage	Est-ce que l'expérience acquise par les agents au cours de la simulation fait évoluer leurs décisions ? Ont-ils une capacité d'apprentissage ? Comment ces mécanismes d'apprentissage sont-ils mis en œuvre ?
Prédiction	Les agents peuvent-ils évaluer les conséquences d'une décision qu'ils pourraient prendre ? Comment prédisent-ils l'effet de leur décision ? En ont-ils la capacité ?
Perception	A quelles informations ont accès les agents ? Quelles sont les variables d'état qu'ils perçoivent ou qu'ils reçoivent d'autres agents (variables internes ou relatives à l'environnement) ?
Interaction	Quelles sont les interactions directes ou indirectes intégrées dans le modèle ? Sur quels mécanismes présents dans la réalité ces interactions sont-elles fondées ? S'agit-il d'interactions locales ou globales ? Est-ce que le modèle permet aux agents de communiquer ? Sous quelle forme ?
Stochasticité	Quelles sont les processus ou variables du modèle qui introduisent de l'aléa ? Pour quelles raisons cet aléa est-il représenté ?
Coopération / agrégation	Existe-t-il un niveau d'organisation composé de groupes d'agents dans le modèle ? Ces groupes sont-ils issus d'un phénomène d'émergence ou définis explicitement car partageant des propriétés communes (notion de *breeds* sous NetLogo) ?
Observation	Quels sont les indicateurs observés au cours de la simulation pour comprendre et analyser le comportement du modèle ? Quelles sont les sorties (données, graphiques)?

Figure 2.16. *Les questions-clés auxquelles ODD invite à répondre afin de spécifier les éléments de conception du modèle agent (d'après [GRI 06, GRI 10])*

? Est-ce que les conditions d'initialisation sont constantes et fondées sur des données de référence ou fixées de manière stochastique ? Il s'agit ici de fournir toutes les informations permettant de reproduire les résultats d'une simulation.

2.4.2.4.2. Données d'entrée (*Input data*)

Les dynamiques représentées dans le modèle peuvent s'appuyer sur des données auxiliaires (par exemple : un environnement spatial prédéfini importé), être modulées par des facteurs de forçage (exemples : le relief, la température de surface, la quantité de précipitations) ou intégrer un modèle existant (exemple : courbe de croissance d'une espèce végétale). Si c'est le cas, c'est à ce niveau qu'il convient de le mentionner. Dans le cas contraire, le protocole préconise d'indiquer explicitement que le modèle ne fait pas appel à des données d'entrée particulières. À noter que cet élément de documentation ne concerne pas les valeurs des paramètres ou des variables d'états pouvant elles aussi provenir de fichiers externes.

2.4.2.4.3. Sous-modèles (*Submodels*)

Les processus composant le modèle et leur ordre d'exécution ont été spécifiés dans la première partie d'ODD. Ici, il convient de détailler précisément le fonctionnement

de chacun d'entre eux, d'indiquer les raisons pour lesquelles certaines hypothèses ont été adoptées, et comment ces sous-modèles ont été calibrés, avec leurs limites éventuelles d'utilisation. Les sous-modèles peuvent être constitués d'équations, d'algorithmes ou de règles spécifiques pour lesquels tous les paramètres doivent être explicités et justifiés. Si les sous-modèles s'appuient sur des théories ou méthodes déjà publiées, il s'agit également d'y faire référence.

2.4.3. *Documentation ODD de l'exemple fil rouge*

Afin d'illustrer l'usage du protocole ODD, nous présentons ci-dessous le modèle « fil rouge » documenté en adoptant ce standard.

2.4.3.1. *Vue d'ensemble (Overview)*

2.4.3.1.1. Objectif (*Purpose*)

L'objectif du modèle de l'exemple fil rouge est d'explorer les conditions de propagation du paludisme dans la sous-région du Maroua (Cameroun) en fonction des mobilités pendulaires des cultivateurs qui réalisent quotidiennement un déplacement entre leur domicile (en ville) et leurs parcelles de culture (à l'extérieur de la ville). Il s'agit de comprendre et de mesurer l'impact de ces mobilités sur la dispersion de la maladie.

2.4.3.1.2. Entités, variables d'états, échelle (*Entities, state variables, and scale*)

Le modèle est composé de deux catégories d'agents : les humains (cultivateurs) et les moustiques. Ces deux types d'entités sont caractérisés par un état infectieux (`est-infecte?`) et un attribut permettant d'indiquer si l'agent est source ou non de l'épidémie (`est-infection-exterieur?`). Chaque agent est également localisé dans l'espace (attributs `xcor` et `ycor`). Les agents humains connaissent par ailleurs le lieu de leur travail et de leur habitation (attributs `maison` et `travail`) en plus de l'heure de début et fin de travail (attributs `début-travail` et `fin-travail`).

L'environnement dans lequel sont situés les agents est composé d'un ensemble de lieux (unités spatiales) caractérisés par une occupation du sol (attribut `nature` de la classe Lieu) associée à une couleur (attribut `pcolor`). Il est précisé pour chaque lieu s'il s'agit d'un lieu d'habitation ou non (attribut `lieu-habitation?`). Il correspond à une cellule de 50m x 50m formant une grille rectangulaire représentant le territoire (dimension : 523 x 424 cellules). Ces dimensions sont imposées par la carte prise en entrée (cf. partie Initialisation de la description ODD).

Concernant la résolution temporelle, le pas de temps de la simulation est fixé à 10 minutes (soit 240 pas de simulation pour une journée). Le modèle est prévu pour effectuer des simulations jusqu'à 30 jours.

Plusieurs variables sont également définies comme paramètres du modèle : le nombre de moustiques présents dans l'environnement ((`nombre-moustiques`), le nombre de cultivateurs (`nombre-humains`), la distance à partir de laquelle un moustique peut atteindre un cultivateur pour le piquer (`distance-contamination`), et la distance domicile-travail de chaque cultivateur (`distance-travail-maison`). Le modèle prévoit par ailleurs d'activer ou non la reproduction des moustiques (`reproduction-moustiques?`) ainsi que la contagion possible entre cultivateurs et moustiques durant le trajet domicile-travail (`contagion-transport?`).

La structure du modèle est présentée de manière simplifiée dans le diagramme de classes UML (figure 2.2). L'ensemble des variables d'états est repris dans le tableau 2.2.

Entité	**Nom variable**	**Valeurs possibles**
Humain & Moustique	`xcor`	[0,523]
	`ycor`	[0,424]
	`est-infecte?`	{vrai,faux}
	`est-infection-exterieur?`	{vrai,faux}
Humain	`maison`	un patch
	`travail`	un patch
	`debut-travail`	[0,240]
	`fin-travail`	[debut-travail,240]
patch	`pcolor`	[0,240]
	`lieu-habitation?`	{vrai,faux}
global	`pas-temps`	[0,2400]
Paramètres	`nombre-moustiques`	[0,1000]
	`nombre-humains`	[0,2000]
	`distance-contamination`	[0,10]
	`distance-travail-maison`	[1,500]
patch	`contagion-transport?`	{vrai, faux}
	`reproduction-moustiques?`	{vrai,faux}

Tableau 2.2. *Résumé des différentes variables d'états et paramètres*

2.4.3.1.3. Processus et ordonnancement (*Process and scheduling*)

À chaque pas de temps, les processus sont exécutés dans l'ordre suivant : 1) les agents moustiques agissent en se déplaçant aléatoirement dans l'espace et en piquant un humain présent dans le rayon de contamination (procédure `go-moustiques`), 2) en fonction de l'heure de la journée, les humains se rendent au travail (procédure `aller-travail`), y restent, et rentrent ensuite à leur domicile en fin de journée (procédure `aller-domicile`). Ces actions des cultivateurs sont intégrées dans la

procédure `go-humains`. Au sein de chaque ensemble d'agents, la plateforme de simulation choisit l'ordre dans lequel les agents exécutent leurs tâches. A la fin de chaque pas de temps, le simulateur met à jour les différents indicateurs de sortie : nombre de moustiques et de cultivateurs infectés en fonction du temps.

La dynamique des patches se limite à la création de nouveaux moustiques (`go-patches`).

2.4.3.2. *Eléments de conception (Design concepts)*

2.4.3.2.1. Principe

On fait l'hypothèse que les agents humains ont une mobilité pendulaire : au cours d'une journée, ils commencent à se rendre à leur lieu de travail, restent immobiles sur place pour travailler puis, rentrent à leur lieu d'habitation. De plus, différentes études ont montré que les moustiques sont présents partout sur le territoire et que leur déplacement restait très local. C'est pourquoi l'hypothèse de considérer qu'ils ont un déplacement aléatoire dans un rayon limité est réaliste.

2.4.3.2.2. Émergence

On observe un phénomène de propagation spatiale de la maladie qui suit les déplacements des agents humains.

2.4.3.2.3. Perception

Les moustiques ont la capacité de percevoir les cultivateurs à proximité (c'est-à-dire, ceux présents dans un rayon de voisinage inférieur au paramètre `distance-contamination`). Les cultivateurs ont quant à eux conscience du temps, ce qui leur permet de décider s'il est l'heure de se rendre au travail, d'y rester, ou de rentrer à la maison. En revanche, ils ne peuvent pas percevoir les autres cultivateurs ni les moustiques.

Les cellules savent si des moustiques sont localisés sur elles (afin de déterminer s'ils produisent ou non un nouvel agent moustique).

2.4.3.2.4. Interaction

Les seules interactions présentes dans le modèle sont les interactions entre un agent humain et un agent moustique. Quand un moustique perçoit un cultivateur dans sa zone de contamination, il va le piquer. Si l'un des deux agents est porteur du paludisme, l'autre sera infecté. Si les deux agents sont dans le même état épidémiologique, la piqûre n'aura aucun effet.

2.4.3.2.5. Stochasticité

Le modèle contient une part de stochasticité à la fois dans l'initialisation et dans la dynamique. En particulier, la position des moustiques dans l'espace (resp. des humains) est aléatoire sur une des cellules (resp. sur un des lieux d'habitation). La position du lieu de travail des humains est également aléatoire parmi les cellules qui ne sont pas des zones d'habitations. Le premier moustique infecté est également choisi aléatoirement. Au cours de la simulation, deux dynamiques intègrent une part d'aléa. Lors de la création d'un nouveau moustique, il est placé sur une des cellules voisines (choisie aléatoirement) de la cellule humide qui « produit » le moustique. De plus, le déplacement d'un moustique contient une part d'aléatoire : il se déplace d'une distance constante mais dans une direction tirée aléatoirement.

2.4.3.2.6. Observation

L'affichage principal du simulateur est une carte affichant les types d'occupation de terrains (cellules noires pour les lieux d'habitations, bleues pour les zones humides et blanches pour les zones de culture), ainsi que les moustiques et les humains. Cet affichage est mis à jour à chaque pas de temps, permettant de visualiser la dynamique des déplacements. De plus, le graphe des interactions passées homme-moustique (des piqures) est également affiché et mis à jour à chaque pas de temps.

Au cours de la simulation, on observe l'évolution du nombre de moustiques et d'humains infectés à l'aide d'une courbe représentant ces valeurs en fonction du temps.

Le simulateur peut calculer et afficher également (lorsque l'utilisateur le demande) le nombre moyen d'interactions homme-moustique (c'est-à-dire le nombre de piqûres de moustiques) qui ont eu lieu avant qu'un humain soit infecté.

2.4.3.3. *Détails (Details)*

2.4.3.3.1. Initialisation (*Initialization*)

La première étape de l'initialisation de la simulation est le chargement de la carte, ce qui fixe la couleur (et donc le type d'utilisation du sol) des différentes cellules. Ensuite des moustiques sont créés aléatoirement dans tout l'espace avec un statut infectieux sain (`est-infecte?` et `est-infection-exterieur?` sont initialisés à faux). Les agents humains sont ensuite créés aléatoirement sur les zones d'habitation, avec un lieu de travail choisi aléatoirement dans les zones de non-habitation et un état infectieux sain. Finalement, une épidémie est créée : un des moustiques est choisi aléatoirement et devient infecté. Il est également noté comme étant la source de l'épidémie (`est-infection-exterieur?` est fixé à vrai).

2.4.3.3.2. Données d'entrée (*Input data*)

Le modèle prend en entrée une carte d'utilisation du sol de la sous-région de Maroua, sous la forme d'une image raster. On n'a représenté sur cette carte que trois

types d'utilisation du sol : lieu d'habitation (en noir), zone humide (en bleu) et zone de culture (en blanc).

2.4.3.3.3. Sous-modèles (*Submodels*)

Nous pouvons distinguer dans le modèle trois sous-modèles correspondants aux trois dynamiques principales du modèles.

Mobilité pendulaire. Le comportement des agents humains est limité à leur mobilité pendulaire : chaque jour, ils se rendent de leur maison à leur travail le matin, travaillent (donc restent immobiles) toute la journée puis rentrent à leur domicile le soir. Le choix est fait en fonction du pas de simulation courant de la journée (c'est-à-dire, en fonction du pas de simulation). Nous avons raffiné ce modèle pour permettre une infection durant le déplacement (paramètre `contagion-transport?`) : lorsque `contagion-transport?` est faux, pour éviter tout contact moustique-humain au cours du déplacement, nous considérons que le déplacement des humains est instantané (alors qu'il prend plusieurs pas de simulation lorsque `contagion-transport?` est vrai).

Démographie des moustiques. Nous avons introduit une dynamique très simple de reproduction des moustiques fondée sur le fait que les moustiques pondent des œufs dans l'eau et vivent le début de leur vie (œuf, larve, nymphe) en milieu aquatique : à chaque jour (donc tous les 240 pas de simulation), les cellules humides (celles dont la couleur est bleue) vont créer un agent moustique si un autre agent moustique est présent sur cette cellule.

Infection. A chaque pas de simulation, tous les agents humains se trouvant dans le rayon de contamination d'un moustique sont piqués. Pour chaque interaction moustique-humain, si l'état infectieux de l'un des deux est infecté et l'autre sain, l'agent sain devient infecté. Dans le cas où ils sont tous les deux sains ou infectés, leur état infectieux reste inchangé.

2.5. Discussion sur la documentation

Le modélisateur dispose aujourd'hui de nombreux langages pour formaliser et documenter son système multi-agent.

Pour ce qui concerne la description de la structure et du comportement du modèle, le langage UML est bien approprié. Il reste le plus répandu et offre une sémantique suffisamment riche pour décrire de nombreux systèmes multi-agents. Dans le cas de modèles intégrant des organisations ou des actions plus complexes, l'utilisation d'AML qui enrichit la notation UML peut s'avérer particulièrement bénéfique. Dans tous les cas, un SMA devrait être décrit à minima par un diagramme de classes,

décrivant sa structure, et par des diagrammes d'activités ou d'états-transitions, modélisant le comportement des entités.

Pour ce qui concerne la documentation, le protocole ODD devient de plus en plus utilisé par les modélisateurs, en particulier pour présenter des modèles multi-agents dans la littérature scientifique. Il devient progressivement un standard reconnu, au même titre qu'UML. On peut également noter que les descriptions ODD incluent de plus en plus des diagrammes de classes UML pour expliciter de manière graphique et synthétique la structure des modèles présentés, ce qui montre la complémentarité des outils. ODD offre la structure de documentation prédéfinie (le contenant) tandis qu'UML permet de produire une partie du contenu sous forme de diagrammes.

Cet effort de formalisation et de documentation du modèle de la part du concepteur est essentiel dans le cycle de vie du SMA. Il permet d'utiliser le modèle de manière plus fiable et éclairée, afin de mieux comprendre les questions auxquelles il tente d'apporter des éléments de réponse et de préciser la structure et le fonctionnement des représentations utilisées pour parvenir à ces fins. Il reste cependant beaucoup à faire dans le domaine, non plus de la conception, mais de l'utilisation des modèles, d'une part pour en fiabiliser l'utilisation, d'autre part pour en faciliter une utilisation éclairée :

– fiabiliser l'utilisation des modèles : il s'agit de passer d'une documentation d'un modèle en tant que résultat d'un processus de modélisation, à une documentation du processus de modélisation lui-même, incluant les justifications des choix opérés et les méthodologies de test utilisées pour assurer autant que faire se peut que l'implémentation informatique du modèle soit fidèle au modèle conceptuel développé; cet aspect inclut également une bonne documentation des conditions d'utilisation d'un modèle afin qu'il soit utilisé à bon escient dans le cadre de la question ciblée pour laquelle il a été développé et éviter les risques d'usage abusif pour résoudre des problèmes pour lesquels il n'a pas été conçu ;

– faciliter l'utilisation des modèles : il devient également important de faciliter la prise en main et l'analyse des résultats issus d'un modèle, par exemple par la définition de différents scénarios d'utilisation et l'analyse détaillée des résultats obtenus.

Des travaux récents montrent que cette question commence à être abordée, par exemple avec le développement de l'outil TRACE (*TRAnsparent and Comprehensive Ecological modelling documentation*) [GRI 14] proposant une standardisation de la documentation associée aux objectifs des modèles, à leur conception et à leur mise en œuvre.

Il est plus que jamais important de favoriser le rapprochement entre les concepteurs et les utilisateurs de modèles dans une démarche d'accompagnement mutuel.

[BAR 07] BARRETEAU O., LE PAGE C., PEREZ P., "Simulation and gaming in natural resource management", *Simulation and Gaming*, vol. 38, n° 2, p. 181-184, 2007.

[BAR 10] BARSEGHIAN D., ALTINTAS I., JONES M. B., CRAWL D., POTTER N., GALLAGHER J., CORNILLON P., SCHILDHAUER M., BORER E. T., SEABLOOM E. W., HOSSEINI P. R., "Workflows and extensions to the Kepler scientific workflow system to support environmental sensor data access and analysis. Barseghian, Derik and Altintas", *Ecological Informatics*, vol. 5, n° 1, p. 42-50, 2010.

[BAT 76] BATTY M., *Urban Modelling: Algorithms, Calibrations, Predictions*, Martin, Leslie and March, Lionel Editors, Cambridge University Press, 1976.

[BAU 01a] BAUER B., MÜLLER J.-P., ODELL J., "Agent UML: a formalism for specifying multiagent software systems", *First international workshop, AOSE 2000 on Agent-oriented software engineering*, Secaucus, NJ, USA, Springer-Verlag New York, Inc., p. 91–103, 2001.

[BAU 01b] BAUER B., MÜLLER J. P., ODELL J., "Agent UML: A Formalism for Specifying Multiagent Interaction", CIANCARINI P., WOOLDRIDGE M., Eds., *Agent-Oriented Software Engineering*, Springer, p. 91-103, 2001.

[BEC 03] BECU N., BOUSQUET F., BARRETEAU O., PEREZ P., WALKER A., "A Methodology for Eliciting and Modelling Stakeholders' Representations with Agent Based Modelling", HALES D., EDMONDS B., NORLING E., ROUCHIER J., Eds., *Proc. of Multi-Agent-Based Simulation III*, Springer, Heidelberg, p. 131-148, 2003.

[BEC 08] BECU N., NEEF A., SCHREINEMACHERS P., SANGKAPITUX C., "Participatory computer simulation to support collective decision-making: Potential and limits of stakeholder involvement", *Land Use Policy*, vol. 25, n° 4, 2008.

[BEC 10] BECU N., BOMMEL P., BOTTA A., LE PAGE C., PEREZ P., "Téchnologies mobilisées pour l'accompagnement", ETIENNE M., Ed., *La modélisation d'accompagnement : une démarche participative en appui au développement durable*, Versailles, France, Quae Edition, p. 183-201, 2010.

[BER 05] BERNON C., COSSENTINO M., PAVON J., "An Overview of Current Trends in European AOSE Research", *Informatica*, vol. 29, p. 379-390, 2005.

[BLI 05] BLIKSTEIN P., ABRAHAMSON D., WILENSKY U., "Netlogo: Where we are, where we're going", EISENBERG M., EISENBERG A., Eds., *unknown*, Boulder, Colorado, IDC, 2005.

[BOM 09] BOMMEL P., Définition d'un cadre méthodologique pour la conception de modéles multi-agents adaptée á la gestion des ressources renouvelables, PhD thesis, Université Montpellier II-Sciences et Techniques du Languedoc, Montpellier, France, 2009.

[BON 03] BON G. L., *Psychologie des foules*, Presses Universitaires de France - PUF, décembre 2003.

[BOO 91] BOOCH G., *Object Oriented Design with Application*, Benjamin Cummings, 1991.

[BOU 98] BOUSQUET F., BAKAM I., PROTON H., LE PAGE C., "Cormas: Common-pool resources and multi-agent systems", *Tasks and Methods in Applied Artificial Intelligence*, p. 826–837, Springer Berlin Heidelberg, 1998.

[BRA 10] BRAX N., AMBLARD F., BECU N., SANTONI L., THIRIOT S., "When predictive modelling meet participatory simulation : a feedback on potential and issues of a combined approach", ULM E., Ed., *unknown*, Paris, France, MAPS2 : Teaching of/with Agent-Based Models in the Social Sciences, 2010.

[BRE 04] BRESCIANI P., PERINI A., GIORGINI P., GIUNCHIGLIA F., MYLOPOULOS J., "TROPOS: An Agent-Oriented Software Development Methodology", *Journal of Autonomous Agents and Multi-Agents Systems*, vol. 8, n° 3, p. 203-236, 2004.

[BRI 00] BRIASSOULIS H., *Analysis of Land Use Change: Theoretical and Modeling Approaches*, Regional Research Institute, West Virginia University, 2000.

[CAI 13] CAILLAULT S., DELMOTTE S., KÊDOWIDÉ C., MIALHE F., VANNIER C., AMBLARD F., BÉCU N., GAUTREAU P., ETIENNE M., HOUET T., "Assessing the influence of social and economical networks on land use and land cover changes: a neutral model based approach", *Environmental Modelling and Software*, vol. 45, p. 64-73, 2013.

[CER 07a] CERVENKA R., TRENCANSKY I., *The Agent Modeling Language - AML: A Comprehensive Approach to Modeling Multi-Agent Systems (Whitestein Series in Software Agent Technologies and Autonomic Computing)*, Birkhäuser Basel, 2007.

[CER 07b] CERVENKA R., TRENCANSKY I., *The Agent Modeling Language - AML: A Comprehensive Approach to Modeling Multi-Agent Systems*, Whitestein Series in Software Agent Technologies and Autonomic Computing, Birkhäuser, 2007.

[CHA 09] CHARLES M. MACAL M. J. N., "AGENT-BASED MODELING AND SIMULATION", Winter Simulation Conference, p. 86-98, 2009.

[CHA 13] CHASSET P., "RK4: Runge-Kutta 4th Order Method for the simulation software NetLogo", `http://flow.chasset.net/netlogo-rk4/`, 2013.

[CHI 13] CHIPEAUX S., Génération automatique de Systèmes Multi-Agents à partir de modèles pour la simulation à large échelle de systèmes complexes de grande taille, PhD thesis, Université de Franche-Comté, Décembre 2013.

[COA 97] COAD P., NORTH D., MAYFIELD M., *Object models: strategies, patterns, and applications*, Yourdon Press, 1997.

[COM 05] COMMOD C., "La modélisation comme outil d'accompagnement", *Natures Sciences Sociétés*, vol. 13, p. 165-168, EDP Sciences, 2005.

[COQ 96] COQUILLARD P., HILL D. R. C., FRONTIER S., *Modelisation et simulation d'ecosystemes des modeles deterministes aux simulations Ã evenements discrets*, Masson, Paris; Milan; Barcelone, 1996.

[DAM 94] DAMASIO A. R., *Descartes' error: emotion, reason, and the human brain*, G.P. Putnam, 1994.

[DAR 03] DARÉ W., BARRETEAU O., "A role-playing game in irrigated system negotiation: between play and reality", *Journal of Artificial Societies and Social Simulation*, vol. 6, n° 3, 2003.

[DEM 95] DEMAZEAU Y., "From Interactions to Collective Behaviour in Agent-Based Systems", *The first European Conference on Cognitive Science*, Saint-Malo, p. 117-132, 1995.

[DEM 97] DEMAZEAU Y., "Steps toward Multi-Agent Oriented Programming", *First in internationnal Workshop on Multi-Agent Systems IWMAS'97*, USA, 1997.

[DEM 03] DEMAZEAU Y., "Créativité Emergente Centrée Utilisateur", *11èmes Journées Francophones sur les Systèmes Multi-Agents*, Hammamet, Hermès, p. 31-36, 2003.

[DRO 13] DROGOUL A., AMOUROUX E., CAILLOU P., GAUDOU B., GRIGNARD A., MARILLEAU N., TAILLANDIER P., VAVASSEUR M., VO D.-A., ZUCKER J.-D., "GAMA: multi-level and complex environment for agent-based models and simulations", GINI M. L., SHEHORY O., ITO T., JONKER C. M., Eds., *AAMAS*, IFAAMAS, p. 1361-1362, 2013.

[EDM 04] EDMONDS B., MOSS S., "From KISS to KIDS - An 'Anti-simplistic' Modelling Approach", DAVIDSSON P., LOGAN B., TAKADAMA K., Eds., *MABS*, vol. 3415 de *Lecture Notes in Computer Science*, Springer, p. 130-144, 2004.

[ETI 10] ETIENNE M., *La modélisation d'accompagnement : une démarche participative en appui au développement durable*, Quae Editions, 2010.

[ETI 11] ETIENNE M., "Pédagogie active et enseignement de la biodiversité par la modélisation d'accompagnement", *Actes du Colloque: Education au développement durable et à la biodiversité : concepts, questions vives, outils et pratiques*, Digne-les-Bains, France, Education au développement durable et à la biodiversité : concepts, questions vives, outils et pratiques, 2011.

[FAI 13] FAIVRE R., IOOSS B., MAHÉVAS S., MAKOWSKI D., MONOD H., *Analyse de sensibilité et exploration de modèles: Application aux sciences de la nature et de l'environnement*, Collection Savoir-faire, Quae éditions, 2013.

[FAP 00] FAP, FIPA KIF Content Language Specification, Rapport, FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS, Suisse, 2000.

[FER 95] FERBER J., *Les systèmes multi-agents : vers une intelligence collective*, Informatique, Intelligence Artificielle, Interéditions, 1995.

[FER 98] FERBER J., GUTKNECHT O., "Aalaadin: a meta-model for the analysis and design of organizations in multi-agent systems", (ED) Y. D., Ed., *ICMAS'98 (International Conference on Multi-Agent Systems)*, IEEE Press, p. 128-135, July 1998.

[FER 03] FERBER J., GUTKNECHT O., MICHEL F., "From Agents to Organizations: an Organizational View of Multi-Agent Systems", P. GIORGINI J. MÜLLER J. O., Ed., *Agent-Oriented Software Engineering IV 4th International Workshop*, Melbourne, Australia, p. 214-230, July 2003.

[FER 04] FERBER J., MICHEL F., BARRANCO J., "AGRE: Integrating Environments with Organizations", *Environments for Multi-Agent Systems, First International Workshop*, Springer, p. 48-56, 2004.

[FIN 94] FININ T., FRITZSON R., MCKAY D., MCENTIRE R., "KQML as an agent communication language", *CIKM '94: Proceedings of the third international conference on Information and knowledge management*, New York, NY, USA, ACM, p. 456-463, 1994.

[FOR 61] FORRESTER J. W., *Industrial Dynamics*, Productivity Pr, student edition édition, 1961.

[FOR 68] FORRESTER J. W., *Principles of Systems*, Pegasus Communications, 1968.

[FOR 69] FORRESTER J. W., *Urban Dynamics*, MIT Press, février 1969.

[FOU 05] FOURNIER S., Intégration de la dimension spatiale au sein d'un modèle multi-agents à base de rôles pour la simulation : Application à la navigation maritime, PhD thesis, Université de Rennes, France, 2005.

[Fou02] Foundation for Intelligent Physical Agents, Genève, Suisse, FIPA Communicative Act Library Specification, 2002.

[FRI 86] FRIJDA N. H., *The emotions*, N° novembre 2003Studies in Emotion and Social Interactions, Cambridge University Press, 1986.

[GAU 11] GAUDOU B., MARILLEAU N., HO T. V., "Toward a Methodology of Collaborative Modeling and Simulation of Complex Systems", *Intelligent Networking, Collaborative Systems and Applications*, p. 27-53, Springer, 2011.

[GLO 04] GLOOR C., STUCKI P., NAGEL K., "Hybrid Techniques for Pedestrian Simulations.", SLOOT P. M. A., CHOPARD B., HOEKSTRA A. G., Eds., *Cellular Automata, 6th International Conference on Cellular Automata for Research and Industry*, vol. 3305 de *Lecture Notes in Computer Science*, Amsterdam, Pays Bas, Springer, p. 581-590, 2004.

[GRI 99] GRIMM V., "Ten years of individual-based modelling in ecology: what have we learned and what could we learn in the future?", *Ecological Modelling*, vol. 115, n° 2-3, p. 129-148, 1999.

[GRI 06] GRIMM V., BERGER U., BASTIANSEN F., ELIASSEN S., GINOT V., GISKE J., GOSS-CUSTARD J., GRAND T., HEINZ S. K., HUSE G., HUTH A., JEPSEN J. U., JORGENSEN C., MOOIJ W. M., MULLER B., PE'ER G., PIOU C., RAILSBACK S. F., ROBBINS A. M., ROBBINS M. M., ROSSMANITH E., RUGER N., STRAND E., SOUISSI S., STILLMAN R. A., VABO R., VISSER U., DEANGELIS D. L., "A standard protocol for describing individual-based and agent-based models", *Ecological Modelling*, vol. 198, n° 1-2, p. 115-126, 2006.

[GRI 10] GRIMM V., BERGER U., DEANGELIS D., POLHILL J., GISKE J., RAILSBACK S., "The ODD protocol: A review and first update", *Ecological Modelling*, vol. 221, p. 2760-2768, 2010.

[GRI 13a] GRIGNARD A., TAILLANDIER P., GAUDOU B., VO D., HUYNH N., DROGOUL A., "GAMA 1.6: Advancing the Art of Complex Agent-Based Modeling and Simulation", *PRIMA 2013: Principles and Practice of Multi-Agent Systems*, vol. 8291 de *Lecture Notes in Computer Science*, p. 117-131, 2013.

[GRI 13b] GRIGNARD A., TAILLANDIER P., GAUDOU B., VO D., HUYNH N., DROGOUL A., "GAMA 1.6: Advancing the Art of Complex Agent-Based Modeling and Simulation", BOELLA G., ELKIND E., SAVARIMUTHU B., DIGNUM F., PURVIS M., Eds., *PRIMA 2013: Principles and Practice of Multi-Agent Systems*, vol. 8291 de *Lecture Notes in Computer Science*, p. 117-131, Springer Berlin Heidelberg, 2013.

[GRI 14] GRIMM V., AUGUSIAK J., FOCKS A., FRANK B. M., GABSI F., JOHNSTON A. S., LIU C., MARTIN B. T., MELI M., RADCHUK V., THORBEK P., RAILSBACK S. F., "Towards better modelling and decision support: Documenting model development, testing,

and analysis using {TRACE}", *Ecological Modelling*, vol. 280, n° 0, p. 129 - 139, 2014, Population Models for Ecological Risk Assessment of Chemicals.

[HAG 73] HAGGETT P., *Analyse spatiale en géographie humaine*, Armand Colin, Paris, 1973.

[HAY 84] HAYNES K., FOTHERINGHAM A., *Gravity and spatial interaction models*, Sage Publications, 1984.

[INT 97] FOR INTELLIGENT PHYSICAL AGENTS F., ""FIPA '97 Specification Part 2: Agent Communication Language", available at http://www.fipa.org", 1997.

[IRW 02] IRWIN E. G., BOCKSTAEL N. E., "Interacting agents, spatial externalities and the evolution of residential land use patterns", *Journal of Economic Geography*, vol. 2, n° 1, p. 31-54, 2002.

[JAG 10] JAGERS H., "Linking data, models and tools: an overview", *International Congress on Environmental Modelling and Software*, Ottawa, Canada, 2010.

[JEA 97] JEAN M. R., PESTY S., "Emergence et SMA", *Intelligence Artificielle et Système Multi-agents, JFIADSMA'97*, La Colle-sur-Loup, France, Hermès, p. 323–342, 1997.

[JEN 00] JENNINGS N., "On agent-based software engineering", *Artificial Intelligence*, vol. 177, n° 2, p. 277-296, 2000.

[JUD 88] JUDGE G., *Introduction to the theory and practice of econometrics*, Wiley, 1988.

[KER 39] KERMACK W. O., MCKENDRICK A. G., "Contributions to the mathematical theory of epidemics", *The Journal of hygiene*, vol. 39, n° 3, p. 271–288, Springer, 1939.

[KER 91] KERMACK W. O., MCKENDRIC A. G., "Contributions to the mathematical theory of epidemics III. Further studies of the problem of endemicity", *Bulletin of Mathematical Biology*, vol. 53, n° 1-2, p. 89–118, mars 1991.

[KLE 07] KLEINBAUM D. G., *Applied regression analysis and multivariable methods*, CengageBrain. com, 2007.

[KOR 08] KORICHI A., BELATTAR B., "Towards a Web Based Simulation Groupware: Experiment with BSCW", *Information Technology Journal*, vol. 7, n° 2, p. 332-337, 2008.

[LAN 89] LANGTON C., *Artificial Life 1*, Addison-Wesley, 1989.

[LAN 13] LANGLOIS P., BLANPAIN B., DAUDÉ E., "MAGéo, une plateforme de simulation multi-agents pour tous", *SimTools*, 2013.

[LEG 97] LEGAY J.-M., *L'expérience et le modèle. Un discours sur la méthode.*, Collection Sciences en questions, INRA Editions, Paris, 1997.

[LEM 90] LE MOIGNE J.-L., *La modélisation des systèmes complexes*, Bordas, 1990.

[LEP 05] LE PAGE C., BOMMEL P., *A methodology for building agent-based simulations of common-pool resources management: from a conceptual model designed with UML to its implementation in CORMAS*, IRRI Press, 2005.

[LUK 04] LUKE S., CIOFFI-REVILLA C., PANAIT L., SULLIVAN K., "Mason: A new multi-agent simulation toolkit", *SwarmFest Workshop*, vol. 8, 2004.

[LUM 04] LUMINET O. V. N., *The Handbook of cognitive psychology*, unknown, 2004.

[MAR 08] MARILLEAU N., CAMBIER C., DROGOUL A., PERRIER E., CHOTTE J., BLANCHART E., "Multiscale MAS modelling to simulate the soil environment: Application to soil ecology", *Simulation Modelling Practice and Theory*, vol. 16, n° 7, p. 736–745, 2008.

[MAS 07] MASSE D., CAMBIER C., BRAUMAN A., SALL S., ASSIGBETSE K., CHOTTE J., "MIOR: an individual-based model for simulating the spatial patterns of soil organic matter microbial decomposition", *European journal of soil science*, vol. 58, p. 1127–1135, 2007.

[MIN 96] MINAR N., Y R. B., Z C. L., The Swarm Simulation System: A Toolkit for Building Multi-Agent Simulations, Rapport, Santa Fe Institute, 1996.

[NAI 10] NAIVINIT W., LE PAGE C., TRÉBUIL G., GAJASENI N., "Participatory agent-based modeling and simulation of rice production and labor migrations in Northeast Thailand", *Environmental Modelling & Software*, vol. 25, n° 11, p. 1345–1358, 2010.

[NGU 09] NGUYEN K. T., BENOIT G., VINH H. T., NICOLAS M., "Application of PAMS Collaboration Platform to Simulation-Based Researches in Soil Science: The Case of the MIcro-ORganism Project", *IEEE-RIVF International Conference on Computing and Telecommunication Technologies*, IEEE-RIVF, 2009.

[NOR 13a] NORTH M., COLLIER N., OZIK J., TATARA E., MACAL C., BRAGEN M., SYDELKO P., "Complex adaptive systems modeling with Repast Simphony", *Complex Adaptive Systems Modeling*, vol. 1, n° 1, Page 3, 2013.

[NOR 13b] NORTH M.J.AND COLLIER N., OZIK J., TATARA E., ALTAWEEL M., MACAL C., BRAGEN M., P. S., "Complex Adaptive Systems Modeling with Repast Simphony", *Complex Adaptive Systems Modeling*, FRG, Springer, 2013.

[OCC 01] OCCELLO M., KONING J.-L., BAEIJIS C., "Conception des Système Multi-Agent : quelques éléments de réflexion méthodologique.", *Technique et science informatique*, vol. 20, n° 2, p. 233-263, 2001.

[ODE 99] ODELL J., PARUNAK H. V. D., BAUER B., "Representing Agent Interaction Protocols in UML", *In OMG Document ad/99-12-01. Intellicorp Inc*, Springer-Verlag, p. 121–140, 1999.

[ODE 00] ODELL J., PARUNAK H., BAUER B., "Extending UML for Agents", G. WAGNER Y. LESPERANCE E. Y., Ed., *Proc. of the Agent-Oriented Information Systems Workshop at the 17th National conference on Artificial Intelligence.*, p. 3–17, 2000.

[OMG 03] OMG, "Unified Modeling Language: Superstructure version 2.0", 2003.

[OST 94] OSTROM E., GARDNER R., WALKER J., *Rules, games, and common-pool resources*, University of Michigan Press, 1994.

[PAD 02] PADGHAM L., WINIKOFF M., "Prometheus: A Methodology for Developing Intelligent Agents", GIUNCHIGLIA F., ODELL J., WEISS G., Eds., *Agent-Oriented Software Engineering III*, vol. 2585 de *LNCS*, Springer-Verlag, p. 174-185, 2002.

[PAG 12] PAGE C. L., BECU N., BOMMEL P., BOUSQUET F., "Participatory Agent-Based Simulation for Renewable Resource Management: The Role of the Cormas Simulation Platform to Nurture a Community of Practice", *J. Artificial Societies and Social Simulation*, vol. 15, n° 1, 2012.

[POL 08] POLHILL J. G., PARKER D., BROWN D., V. G., "Using the ODD protocol for describing three agent-based social simulation models of land-use change", *Journal of Artificial Societies and Social Simulation*, vol. 11, n° 2, 2008.

[POL 10] POLHILL J. G., "ODD Updated", *Journal of Artificial Societies and Social Simulation*, vol. 13, n° 4, 2010.

[PRO 05] PROVITOLO D., "Un exemple d'effet de dominos : la panique dans les catastrophes urbaines", *Cybergeo : Revues européenne de géographie*, vol. 328, 2005.

[PRO 07] PROVITOLO D., "A proposition for a classification of the catastrophe systems based on complexity criteria", *Proceedings of EPNACS'07, Emergent Properties in Natural and Artificial Complex Systems*, 4-5 octobre 2007, 2007.

[PRU 02] PRUNEAU D., LAPOINTE C., "Un, deux, trois, nous irons au bois. L'apprentissage expérientiel et ses applications en éducation relative à l'environnement", *Éducation et francophonie*, vol. 30, n° 2, p. 241-256, 2002.

[QUE 09] QUESNEL G., DUBOZ R., RAMAT E., "The Virtual Laboratory Environment – An operational framework for multi-modelling, simulation and analysis of complex dynamical systems", *Simulation Modelling Practice and Theory*, vol. 17, p. 641-653, April 2009.

[RAI 11] RAILSBACK S. F., GRIMM V., *Agent-based and individual-based modeling : a practical introduction*, Princeton University Press, 2011.

[RAO 91] RAO A. S., GEORGEFF M. P., "Modeling rational agents within a BDI-architecture", *Proceedings of Knowledge Representation and Reasoning*, Morgan Kaufmann Publishers, 1991.

[REN 02] RENNARD J.-P., *Vie Artificielle. Où la biologie rencontre l'informatique*, Vuibert, 2002.

[RES 96] RESNICK M., "StarLogo: an environment for decentralized modeling and decentralized thinking", *Conference companion on Human factors in computing systems*, p. 11–12, 1996.

[RES 97] RESNICK M., *Turtles, Termites and Traffic Jams Explorations in Massively Parallel Microworlds.*, MIT Press, Cambridge, 1997.

[REU 13] REUILLON R., LECLAIRE M., REY-COYREHOURCQ S., "OpenMOLE, a workflow engine specifically tailored for the distributed exploration of simulation models", *Future Generation Computer Systems*, vol. 29, n° 8, p. 1981 - 1990, 2013.

[RYK 96] RYKIEL E., "Testing ecological models: the meaning of validation", *Ecological Modelling*, vol. 90, p. 229-244, 1996.

[SAL 09] SALTELLI A., CHAN K., SCOTT E., *Sensitivity Analysis*, N° n° 2008Wiley paperback series, Wiley, 2009.

[SAN 05] SANDERS L., "Simulation des systèmes urbains", *Ecole thématique CNRS : modélisation et simulation multi-agent de systèmes complexe pour les SHS*, Poquerolles, 2005.

[SCH 65] SCHLAGER K. J., "A LAND USE PLAN DESIGN MODEL", *Journal of the American Institute of Planners*, vol. 31, n° 2, p. 103-111, 1965.

[SCH 99a] SCHERER K. R., "Appraisal theory. Handbook of cognition and emotion.", *unkown*, p. 637–663, unknown, 1999.

[SCH 99b] SCHERER K. R., "On the Sequential Nature of Appraisal Processes: Indirect Evidence from a Recognition Task", *Cognition and Emotion*, vol. 13, n° 6, p. 763–793, 1999.

[SCH 02] SCHWEITZER F., "Brownian Agent Models for Swarm and Chemotactic Interaction. Abstracting and Synthesizing the Principles of Living Systems", *Fifth German Workshop on Artificial L*, Nerlin, Allemagne, p. 181-190, 2002.

[SCH 03] SCHWEITZER F., FARMER J. D., *Brownian Agents and Active Particles: Collective Dynamics in the Natural and Social Sciences*, Springer, 2003.

[SIL 03] SILVA V., GARCIA A., BRANDÃO A., CHAVEZ C., LUCENA C., ALENCAR P., "Taming Agents and Objects in Software Engineering", *Software Engineering for Large-Scale Multi-Agent Systems: Research Issues and Practical Applications, volume LNCS 2603*, Springer-Verlag, p. 1–25, 2003.

[SIN 00] SINCLAIR T., SELIGMAN N., "Criteria for publishing papers on crop modeling", *Field Crops Research*, vol. 68, n° 3, p. 165-172, 2000.

[STU 03] STURM A., DORI D., SHEHORY O., "Single-model method for specifying multi-agent systems", *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, New York, NY, USA, ACM, p. 121–128, 2003.

[TAY 07] TAYLOR I. J., DEELMAN E., GANNON D. B., SHIELDS M., *Workflows for e-Science: Scientific Workflows for Grids*, Springer, 2007.

[TIS 04] TISUE S., WILENSKY U., "NetLogo: A simple environment for modeling complexity", *International Conference on Complex Systems*, p. 16–21, 2004.

[TRE 05] TRENCANSKY I., CERVENKA R., "Agent Modeling Language (AML): A Comprehensive Approach to Modeling MAS", *Informatica*, vol. 29, p. 391-400, 2005.

[TRO 09] TRONG KHANH NGUYEN B. GAUDOU T. V. H., MARILLEAU N., "PAMS Collaboration Platform to Simulation-Based Researches in Soil Science: The Case of the MIcro-ORganism Project", *RIVF*, International Conference on Computing and Communication Technologies, IEEE, p. 1-8, 2009.

[UML05] UML 2.0 Superstructure Specification, Rapport, Object Management Group (OMG), August 2005.

[VAR 13] VARENNE F., SILBERSTEIN M., *Modéliser & simuler. Epistémologies et pratiques de la modélisation et de la simulation, tome 1*, Editions Matériologiques, 2013.

[VIA 11] VIAL G., "Le système proie-prédateur de Volterra-Lotka.", *Site de l'Ecole Centrale de Lyon*, 2011.

[VIN 99] VINCK D., "Les objets intermédiaires dans les réseaux de coopération scientifique", *Revue Française de Sociologie*, vol. 40, n° 2, p. 385-414, 1999.

[VOI 10] VOINOV A., BOUSQUET F., "Modelling with stakeholders", *Environmental Modelling & Software*, vol. 25, n° 1, Faculty of Geo-Information Science and Earth Observation (ITC), University of Twente and Cirad, UPR Green, 2010.

[WAL 77] WALLISER B., *Systèmes et modèles. Introduction critique à l'analyse des systèmes.*, Editions du Seuil, 1977.

[WAN 09] WANG L., JIE W., CHEN J., *Grid Computing: Infrastructure, Service, and Applications*, CRC Press, 2009.

[WAT 92] WATSON D., CLARK L. A., "On traits and temperament: general and specific factors of emotional experience and their relation to the five-factor model", *Journal of Personality*, vol. 60, n° 2, p. 441–476, juin 1992, PMID: 1635050.

[WIL 74] WILSON A., *Urban and regional models in geography and planning*, Wiley, 1974.

[WOO 97] WOOLDRIDGE M., "Agent based software engineering", *Software Engineering*, vol. 144, p. 26–37, 1997.

[z83] "Fatal Panic on the Bridge, Twelve persones killed and many injured", `http://chroniclingamerica.loc.gov/lccn/sn83030214/1883-05-31/ed-1/seq-1/`, mai , 31 1883.

[za83] "Terrible Disaster", `http://chroniclingamerica.loc.gov/lccn/sn82014381/1883-05-31/ed-1/seq-1/`, mai , 31 1883.

[ZAM 03] ZAMBONELLI F., JENNINGS N., WOOLDRIDGE M., "Developing Multiagent Systems: The Gaia Methodology", *ACM Trans. on Software Engineering and Methodology*, vol. 12, n° 3, p. 317-370, 2003.