Un modèle d'environnement pour la simulation multiniveau - Application à la simulation de foules

Jonathan Demange

SPIM

Thèse de Doctorat

école doctorale **sciences pour l'ingénieur et microtechniques**

UNIVERSITÉ DE TECHNOLOGIE BELFORT-MONTBÉLIARD

Un modèle d'environnement pour la simulation multiniveau

Application à la simulation de foules

Jonathan DEMANGE

SPIM

Thèse de Doctorat

école doctorale **sciences pour l'ingénieur et microtechniques**

UNIVERSITÉ DE TECHNOLOGIE BELFORT-MONTBÉLIARD

N° 1 9 4

THÈSE présentée par

Jonathan DEMANGE

pour obtenir le

Grade de Docteur de
l'Université de Technologie de Belfort-Montbéliard

Spécialité : **Informatique**

Un modèle d'environnement pour la simulation multiniveau

Application à la simulation de foules

Unité de Recherche :
Laboratoire Systèmes et Transports (IRTES-SET)

Soutenue le 20 décembre 2012 devant le Jury :

Vincent CHEVRIER	Rapporteur	Maître de Conférences HDR à l'Université Henri Poincarré
René MANDIAU	Rapporteur	Professeur à l'Université de Valenciennes
Christophe NICOLLE	Président	Professeur à l'Université de Bourgogne
Lhassane IDOUMGHAR	Examinateur	Maître de Conférences HDR à l'Université de Haute Alsace
Abderrafiaa KOUKAM	Directeur de thèse	Professeur à l'UTBM
Stéphane GALLAND	Co-directeur de thèse	Maître de Conférences à l'UTBM

Remerciements

Je tiens tout d'abord à remercier Vincent Chevrier et René Mandiau pour avoir accepté d'être mes rapporteurs de thèse. Je les remercie pour leurs remarques pertinentes sur le manuscrit et leur participation à mon jury de thèse.

Je remercie également Christophe Nicolle d'avoir présidé la soutenance ainsi que Lhassane Idoumghar d'avoir accepté de faire partie de mon jury.

Merci à Abderrafiâa Koukam pour sa patience et sa rigueur scientifique sans lesquelles ce manuscrit n'aurait pu aboutir.

Merci à Stéphane Galland qui fut plus qu'un simple co-directeur de thèse. Je le remercie pour sa gentillesse, le temps qu'il a sacrifié pour voir ces travaux aboutir et pour sa grande expérience de développeur qui m'a permis d'en apprendre toujours plus.

Je tiens également à remercier tout le groupe de recherche sur les systèmes multiagents et l'optimisation, et notamment : Nicolas Gaud, Gillian Basso, Vincent Hilaire, Olivier Lamotte, Jean-Michel Contet, Fabrice Lauri, Florian Béhé, Pablo Gruer, Philippe Descamps et Li Shi. Merci pour vos conseils, vos discussions ainsi que pour la bonne humeur dans laquelle s'est effectué le travail.

Un grand merci à mes amis, et tout particulièrement à Pierre, qui ont été à mes côtés tout au long de cette thèse. Merci pour avoir été présents, notamment dans les moments les plus difficiles.

Merci à ma mère, mes grands parents et mes sœurs pour leur présence, leur suivi et leur soutien.

Merci à celles et ceux qui ont partagé mon quotidien au sein du laboratoire Système et Transport et de l'Université de Technologie de Belfort-Montbéliard, et qui m'ont permis de passer d'agréables moments. La liste est longue et je remercie tout particulièrement : Frédéric Lassabe, Sylvaine Schlienger, Wafaa Ait-Cheik-Bihi, Mohamad Dib, Qiao Lassabe, Adnen El Amraoui, Imad Matraji, Jun Hu, Adeel Mehmood, Fayez Shakil Ahmed, Ariane Glatigny, Sébastien Salmon, Frédéric Gilles, Jérôme Mary et Simon Le Lann.

SOMMAIRE

SOMMAIRE DES DÉFINITIONS

I

CONTEXTE ET ÉTAT DE L'ART

1

INTRODUCTION

1.1/ CONTEXTE

La simulation de foules de piétons constitue un enjeu majeur dans de nombreuses applications. Elle a été fortement utilisée dans l'industrie des jeux vidéo et l'animation. À travers le film *« le Seigneur des Anneaux »* , l'industrie cinématographique a exploité la simulation de foules de guerriers pour réaliser des scènes de batailles proches de la réalité. L'intérêt est de réduire les coûts liés à la gestion des figurants.

L'étude des phénomènes de foules suscite aussi un intérêt important dans l'aménagement des sites urbains, l'étude de la sécurité, l'architecture, et l'analyse des flux ; où les deux classes d'applications dominantes sont la simulation d'évacuations et l'analyse des flux des déplacements d'un grand nombre d'individus dans un environnement intérieur ou extérieur [Thalmann et Musse, 2007]. L'objectif de ces simulations est d'aider les décideurs et les experts à comprendre la *« relation entre l'organisation de l'espace et les comportements humains »* [Okazaki et Matshushita, 1993]. Les environnements considérés sont hétéroclites : les rues [Farenc et al., 1999, Thomas et Donikian, 2000], les bâtiments [Braun et al., 2005a, Thompson et E., 1995], les métros [Hareesh, 2000], les bâteaux [Klüpfel et al., 2000], les avions [Owen et al., 1998], les stades [Still, 2000] ou encore les aéroports [Szymanezyk et al., 2011]. Dans la suite de cette thèse, nous nous focaliserons essentiellement sur les environnements intérieurs. Un environnement intérieur est un *environnement physique* [Odell et al., 2002] délimité par des barrières infranchissables (*e.g.* des murs) et disposant de passages situés à ses frontières permettant de gérer les flux entrant et sortant d'entités, tels que les gares, les aéroports, les bâtiments, *etc*.

Les environnements sont généralement étudiés suivant deux principaux points de vue. Le premier concerne l'évolution de la structure de l'environnement [Lamarche et Donikian, 2004, Pouyanne, 2004] et le second traite davantage les activités sociales, et notamment l'analyse du trafic et des déplacements de foules [Farenc et al., 1999, Jian et al., 2005]. L'étude de ces points de vue nécessite la prise en compte non seulement des aspects macroscopiques, mais aussi des entités au niveau microscopique. Quel que soit l'environnement étudié, extérieur ou intérieur, l'approche multiagent est un outil particulièrement bien adapté à l'étude des dynamiques de déplacement de populations. Les systèmes multiagents s'avèrent plus flexibles que les modèles macroscopiques à base d'équations différentielles pour simuler des phénomènes spatiaux et évolutifs [Bretagnolle et al., 2003].

C'est dans ce contexte que se situe notre problématique de thèse. Notre objectif est de

proposer un modèle et des outils permettant de simuler des foules de piétons dans un environnement intérieur.

1.2/ OBJECTIF DE LA THÈSE

L'objectif principal de cette thèse peut être résumé par la formule suivante :

Proposer un modèle organisationnel et holonique, et mettre en œuvre des outils pour la simulation de piétons dans des environnements intérieurs.

Pour concevoir cette simulation, la modélisation de la dynamique d'une population de piétons doit être clairement distinguée de celle de la structure de l'environnement. En effet, la modélisation structurelle de l'espace se rapporte à l'aspect environnemental de la simulation alors que le modèle de la dynamique des piétons concerne davantage le modèle du système.

L'environnement est communément considéré comme l'une des parties essentielles d'une simulation multiagent. Cependant, différentes perspectives existent sur le rôle qu'il joue dans un système multiagent et sur sa définition. Les environnements utilisés dans cette thèse peuvent également être considérés comme des cas particuliers d'*« environnement physique »* [Odell et al., 2002]. La notion d'*« environnement physique »* se réfère à la classe de systèmes dans laquelle les agents, ainsi que les objets, disposent d'une position explicite et produisent des actions elles-aussi localisées [Ferber et Müller, 1996].

1.2.1/ MODÈLE ORGANISATIONNEL ET HOLONIQUE

Une foule de piétons peut être considérée comme un système composé d'un grand nombre d'entités en interaction, dont la dynamique globale ne peut se réduire à la somme des comportements de ses composants. Nous retrouvons dans cette vision toutes les caractéristiques d'un système complexe. Typiquement, ces systèmes exhibent des structures hiérarchiques et des processus d'auto-organisation. La simulation multiniveau fondée sur les modèles multiagents holoniques constitue une approche permettant d'analyser la dynamique de tels systèmes. Elle permet d'analyser le système en considérant plusieurs niveaux d'observation (microscopique, mésoscopique et macroscopique) et prend en compte les ressources de calcul disponibles.

Le concept de holon permet à des agents de se regrouper pour créer un agent de niveau supérieur. Par conséquent, un agent peut aussi se décomposer en plusieurs agents de niveau inférieur. C'est ce processus de composition/décomposition qui contribue à l'élaboration de modèles de simulation permettant d'analyser les systèmes à différents niveaux d'abstraction. Pour élaborer un support pour la modélisation, nous enrichissons le métamodèle CRIO (Capacité, Rôle, Interaction, Organisation) proposé par notre équipe dans [Cossentino et al., 2010, Gaud, 2007].

En s'appuyant sur la méthodologie ASPECS, basée sur le métamodèle CRIO, nous proposons un modèle holonique pour la simulation multiniveau de foules de piétons. Dans cette thèse, l'environnement est modélisé par un système multiagent holonique en charge d'assurer les différentes missions classiquement assignées à l'environnement. L'environnement est composé d'une structure hiérarchique d'objets (zones, sous-zones, *etc.*) et

d'un ensemble d'organisations devant assurer ses différentes missions. Le modèle de l'environnement proposé permet ainsi de faire cohabiter plusieurs niveaux d'abstraction au sein d'une même simulation.

1.2.2/ EXTENSION DE LA PLATEFORME JASIM ET APPLICATION

Afin de concrétiser notre approche de modélisation holonique de l'environnement, nous avons contribué au développement de la plateforme de simulation JASIM. Nous avons particulièrement implanté les organisations et les rôles identifiés dans le modèle holonique de l'environnement.

L'objectif de la plateforme de simulation JASIM est de fournir un environnement logiciel permettant de simuler la dynamique d'une ville, incluant l'intérieur des bâtiments, pour différents modes de déplacement : piétons, véhicules, bus, vélos, *etc*.

La plateforme JASIM est ensuite utilisée pour simuler les déplacements des voyageurs au sein d'un terminal d'aéroport. La simulation des flux aux Postes d'Inspection Filtrage (PIF) a été sélectionnée comme cas d'application de notre modèle holonique de l'environnement, car sa taille et sa complexité permettent de montrer l'intérêt de modèles multiniveaux. L'objectif premier de cette application n'est pas une étude détaillée des flux et des phénomènes de congestion pouvant se produire dans l'ensemble du terminal de l'aéroport. Nous nous focalisons principalement sur les halls d'enregistrement et d'embarquement, ainsi que sur les PIF et leur file d'attente.

1.3/ PLAN DE LA THÈSE

Ce mémoire présente la conception d'un modèle de simulation multiniveau de foules au sein d'un environnement intérieur. Ce modèle est appliqué au problème de la simulation des déplacements de voyageurs dans un terminal d'aéroport. Le mémoire se décompose en deux parties : (i) un état de l'art sur les systèmes multiagents et la simulation de foules et (ii) la modélisation de l'environnement fondée sur la méthodologie ASPECS et sa mise en œuvre à travers la simulation des déplacements de passagers dans un terminal d'aéroport.

La première partie est constituée des deux premiers chapitres. Nous y présentons l'état de l'art sur les systèmes multiagents holoniques ainsi que sur les modèles de simulation de foules.

Le chapitre 2 introduit les concepts des systèmes multiagents holoniques, présente la méthodologie ASPECS pour la conception de systèmes multiagents et répertorie de manière non exhaustive quelques plateformes d'implantation et de simulation.

Le chapitre 3 est consacré aux modèles de simulation de foules. Après une définition de la simulation multiniveau et de la simulation orientée-agents, nous présentons un état de l'art sur les modèles de simulation de foules.

La seconde partie (chapitres 4, 5 et 6) constitue le cœur de notre contribution. Elle présente les modèles organisationnel et holonique pour la simulation de piétons au sein d'un environnement intérieur, ainsi que son implantation et son déploiement pour la simulation de passagers au sein d'un terminal d'aéroport.

Le chapitre 4 propose un modèle organisationnel de l'environnement fondé sur le métamodèle CRIO. En effet, l'élaboration d'un modèle pour la simulation multiagent doit intégrer non seulement la modélisation des agents mais aussi celle de leur environnement. Dans nos travaux, nous modélisons l'environnement dans lequel sont situés les agents (piétons ou groupes de piétons) par un système multiagent assurant des missions spécifiques. Après la présentation d'une vue globale d'un modèle de simulation multiniveau de foules et des missions de l'environnement, nous proposons une modélisation de l'environnement et de ses différentes composantes. Puis, avant de conclure ce chapitre, nous définissons le modèle organisationnel de l'environnement qui est la base du modèle holonique présenté dans le chapitre suivant.

Le chapitre 5 présente un modèle multiagent d'un environnement pour la simulation de foules en environnement intérieur. Le modèle proposé est basé sur le métamodèle CRIO et sur le modèle organisationnel présenté dans le chapitre précédent. Le chapitre 5 détaille le système multiagent composé de holons en charge de simuler un environnement intérieur. Le modèle multiniveau de l'environnement et de sa dynamique est la principale contribution de ce chapitre. L'environnement est décomposé hiérarchiquement en un ensemble de niveaux. Chaque holon est associé à une partie de l'environnement et détermine le meilleur niveau d'abstraction à simuler suivant un ensemble d'indicateurs.

Le chapitre 6 illustre notre contribution en considérant une application de simulation de voyageurs au sein d'un terminal d'aéroport. Nous décrivons le scénario de simulation considéré. Par la suite, nous présentons un modèle concret qui est déployé sur les plateformes JASIM et JANUS. Ce chapitre se conclut par la présentation d'expérimentations décrivant, d'une part, quelques résultats concernant les temps d'attente des voyageurs aux PIF, et d'autre part, une analyse des performances de calcul des modules de JASIM en relation avec notre modèle holonique.

Enfin, une conclusion présente un bilan de nos travaux de recherche et adresse quelques perspectives. Elles portent principalement sur la simulation multiniveau et l'intégration de l'aspect sémantique dans la modélisation de l'environnement.

2

SYSTÈMES MULTIAGENTS HOLONIQUES : CONCEPTS, MÉTHODOLOGIES ET PLATEFORMES

CETTE thèse propose une approche fondée sur les Systèmes MultiAgents Holoniques (SMAH) pour la conception d'un modèle de simulation de foules dans un environnement intérieur. Pour cela, il nous faut choisir une méthodologie pour faciliter la conception de notre modèle, ainsi que la plateforme supportant l'approche des SMAH. Ce chapitre est structuré en quatre parties. La première présente la définition des concepts propres aux Systèmes MultiAgents (SMA). La seconde partie expose l'approche organisationnelle pour la modélisation de ces systèmes. Elle présente également les SMAH. La troisième partie aborde les méthodologies *« orientées-agents »* et la dernière fait état des plateformes agents.

Sommaire

2.1/ INTRODUCTION

Les Systèmes MultiAgents (SMA) revêtent de plus en plus d'importance pour leur capacité à aborder les systèmes complexes. Le domaine des SMA peut être vu comme le confluent de plusieurs disciplines de recherche : l'intelligence artificielle [Weiss, 1999] pour les aspects décisionnels de l'agent, l'intelligence artificielle distribuée [Fagin, 1995] et plus généralement les systèmes distribués pour les interactions (*cf.* définition 2.12) entre agents et la distribution de la résolution et de l'exécution, et enfin le génie logiciel [Jennings, 2000, Shoham, 1994] pour l'approche de modélisation orientée-agents et la création de composants logiciels autonomes.

L'intérêt de ces systèmes est double. Ils contribuent d'une part à l'analyse des mécanismes produits lorsque des entités autonomes interagissent entre elles. Les SMA permettent alors de modéliser, expliquer et simuler des phénomènes naturels. Ils suscitent également des modèles d'auto-organisation. Par exemple, ils peuvent être utilisés pour modéliser l'ensemble des foyers au sein d'une ville, et ainsi déterminer l'évolution des migrations intra-urbaines [Vanbergue, 2003]. D'autre part, ils sont utilisés pour la réalisation de systèmes complexes via les concepts d'agent, de communication, de coopération et de coordination d'actions [Jarras et Chaib-draa, 2002]. Le travail en collaboration d'un groupe d'individus (*e.g.* des chef de projets, des ingénieurs, des experts, *etc.*) sur un projet est un exemple de système complexe pouvant être modélisé par les SMA.

Ce chapitre se divise en quatre parties. Nous commençons par la définition des concepts propres aux SMA. Ensuite, nous exposons l'approche organisationnelle pour les Systèmes MultiAgents Holoniques (SMAH). Puis, nous présentons les méthodologies *« orientées-agents »* en détaillant celle que nous utilisons pour la conception de notre modèle. Enfin, nous terminons par une comparaison succincte des plateformes agents.

2.2/ DÉFINITIONS ET FONDEMENTS DES SYSTÈMES MULTIAGENTS

Cette section s'attache à fournir un bref récapitulatif des définitions fondamentales au cœur de cette thèse. Les concepts d'agent et de systèmes multiagents y seront notamment introduits.

La plupart des auteurs s'accordent généralement pour définir un SMA comme un système composé d'agents qui communiquent et collaborent pour achever des objectifs spécifiques personnels ou collectifs. La communication implique l'existence d'un espace partagé support de cette communication. Cet espace est généralement qualifié d'Environnement.

[Chaib-Draa et al., 2001] proposent la définition 2.1 d'un SMA, où il est différencié d'un simple regroupement d'agents indépendants (sans interactions). Bien que généraliste, cette définition a le mérite de différencier les SMA d'une simple collection d'agents.

Définition 2.1 Système multiagent [Chaib-Draa et al., 2001]

Un système multiagent se distingue d'une collection d'agents indépendants par le fait que les agents interagissent en vue de réaliser conjointement une tâche ou d'atteindre conjointement un but particulier.

[Ferber, 1995] propose la définition 2.2 d'un SMA en abordant point par point les aspects fondamentaux qui lui sont associés.

Définition 2.2 Système multiagent [Ferber, 1995]

On appelle système multiagent, un système composé des éléments suivants :
- Un environnement E, disposant en général d'une métrique,
- Un ensemble d'objets O, auxquels on peut associer une position dans E à un moment donné. Ces objets (hormis les agents) sont passifs : les agents peuvent les percevoir, les créer, les détruire et les modifier.
- Un ensemble d'agents A, lesquels représentent les entités actives du système,
- Un ensemble de relations R, qui unissent les objets (et agents) entre eux,
- Un ensemble d'opérateurs Op permettant aux agents de A de percevoir, produire, consommer, transformer et manipuler des objets de O,
- Et, des opérateurs chargés de représenter l'application de ces opérations et la réaction du monde à cette tentative de modification, que l'on appellera les lois de l'univers.

Ces définitions nous montrent que les SMA contiennent un ensemble d'agents autonomes en interaction. En comparaison à la définition de [Chaib-Draa et al., 2001], celle de [Ferber, 1995] intègre la notion d'environnement en plus des notions d'agent et d'interaction. Les SMA peuvent résoudre des tâches complexes via des collaborations entre agents ou via un phénomène d'émergence (provoqué par des interactions directes ou indirectes entre agents). Un SMA est une société organisée d'agents dans laquelle un certain nombre de phénomènes peuvent émerger comme la résultante des interactions entre les agents [1]. Cette notion d'émergence est essentielle dans les SMA, car c'est l'une des propriétés qui les rendent si aptes à modéliser les systèmes complexes [Gaud, 2007].

Dans les sections suivantes, nous détaillons les notions les plus significatives présentées dans la définition 2.2 : agent, interaction et environnement.

2.2.1/ Définition de la notion d'agent

La plupart des définitions sur la notion d'agent sont fortement liées à une application ou un domaine d'application [Briot et Demazeau, 2001, Ferber, 1999, Wooldridge et Jennings, 1999]. L'une des définitions les plus célèbres de la notion d'agent a été formulée par [Russell et Norvig, 1995b], ils considèrent un agent comme *« Tout ce qui peut être vu comme percevant son environnement à l'aide de capteurs et agissant sur cet environnement à l'aide d'effecteurs, de façon autonome »* . Cette définition très générale et volontairement minimaliste dans sa formulation (uniquement) a été étendue notamment par Ferber pour, en outre, accentuer l'importance de l'environnement (qui demeurait rarement spécifié) [Ferber, 1995, p. 13] :

1. Cette résultante n'est d'ailleurs pas forcément prédictible.

Définition 2.3 Agent [Ferber, 1995]

Une entité physique ou virtuelle :
- qui est capable d'agir dans un environnement,
- qui peut communiquer directement avec d'autres agents,
- qui est mue par un ensemble de tendances (sous la forme d'objectifs individuels ou d'une fonction de satisfaction, voire de survie, qu'elle cherche à optimiser),
- qui possède des ressources propres,
- qui est capable de percevoir (mais de manière limitée) son environnement,
- qui ne dispose que d'une représentation partielle de cet environnement (et éventuellement aucune),
- qui possède des compétences et offre des services,
- qui peut éventuellement se reproduire,
- dont le comportement tend à satisfaire ses objectifs, en tenant compte des ressources et des compétences dont elle dispose, et en fonction de sa perception, de ses représentations et des communications qu'elle reçoit.

2.2.1.1/ AUTONOMIE DES AGENTS

[Russell et Norvig, 1995a] donnent une définition d'un système autonome (*cf.* définition 2.4). Cette définition englobe celle d'un agent autonome, considéré lui-même comme un système. Ici, l'autonomie est liée au comportement d'un agent. Il est d'ailleurs fait référence à la notion d'expérience, impliquant ainsi un raffinement du concept d'agent.

Définition 2.4 Système autonome [Russell et Norvig, 1995a]

Un système est autonome dans la mesure où son comportement est déterminé par sa propre expérience.

[Maes, 1995] propose une définition de l'autonomie (*cf.* définition 2.5) pour un agent.

Définition 2.5 Agents autonomes [Maes, 1995]

Les agents autonomes sont des systèmes informatiques qui peuplent un environnement complexe et dynamique, perçoivent et agissent de manière autonome dans cet environnement et, ce faisant, réalisent un ensemble d'objectifs ou de tâches pour lesquelles ils sont conçus.

Cette définition n'a pas pour but de remettre en cause la définition d'un agent, mais simplement de mettre en avant la notion d'autonomie qui lui est associée. En son sens, [Maes, 1995] définit l'autonomie d'un agent comme étant sa capacité à prendre une décision suivant les informations qu'il peut avoir via ses perceptions. L'agent peut être influencé, mais jamais contrôlé par une autre entité.

2.2.2/ ENVIRONNEMENT DANS UN SYSTÈME MULTIAGENT

L'environnement est communément défini par tout ce qui entoure un agent. [Weyns et al., 2005] donnent une définition globale de l'environnement (*cf.* définition 2.6), sans pour autant faire référence à un type d'agent particulier.

Définition 2.6 Environnement [Weyns et al., 2005]

L'environnement est une abstraction de premier ordre qui fournit les conditions environnantes aux agents pour exister et qui sert d'intermédiaire à la fois pour les interactions entre agents et l'accès aux ressources

[Odell et al., 2002] définissent trois types d'environnements que sont :
– Les environnements de communication (*cf.* section 2.2.2.1),
– Les environnements sociaux (*cf.* section 2.2.2.2) et
– Les environnements physiques (*cf.* section 2.2.2.3).

2.2.2.1/ ENVIRONNEMENT DE COMMUNICATION

[Odell et al., 2002] proposent la définition 2.7 pour la notion d'environnement de communication.

Définition 2.7 Environnement de communication [Odell et al., 2002]

L'environnement de communication fournit les principes, les processus et les structures qui permettent à une infrastructure ou une plateforme de transporter de l'information entre les agents.

Les environnements de communication servent uniquement de support à la définition des interactions entre les agents. Les interactions constituent une part essentielle à la mise en œuvre des SMA. En effet, elles permettent aux agents de collaborer et de coopérer pour produire des comportements collectifs complexes.

2.2.2.2/ ENVIRONNEMENT SOCIAL

[Odell et al., 2002] proposent la définition 2.8 pour la notion d'environnement social dans un SMA.

Définition 2.8 Environnement social [Odell et al., 2002]

Un environnement social est un environnement de communication dans lequel les agents interagissent de manière coordonnée.

Les environnements sociaux sont donc une extension des environnements de communication. Ils fournissent des protocoles de communications évolués permettant aux agents

d'interagir de manière coordonnée. Ainsi, la communication est un cas particulier d'interaction. Des travaux proposent des protocoles de communications entre agents [Barbuceanu et Fox, 1995, Boissier et Demazeau, 1994, Finin et al., 1994, FIPA, 1998].

2.2.2.3/ ENVIRONNEMENT PHYSIQUE

Les environnements physiques sont des environnements spatiaux dans lesquels des agents sont immergés. Ces agents peuvent se déplacer, percevoir et agir dans ces environnements. [Odell et al., 2002] proposent la définition 2.9 pour la notion d'environnement physique.

Définition 2.9 Environnement physique [Odell et al., 2002]

L'environnement physique fournit les principes et les processus qui régissent et supportent une population d'entités.

Les agents (ou entités autonomes) possèdent un corps correspondant à une représentation physique de l'agent à laquelle est associé son comportement [Michel, 2004]. On dira alors qu'ils sont situés dans l'environnement. Le concept de corps a été raffiné et adapté par [Galland et al., 2009] à la simulation dans des environnements virtuels 3D. Le corps sert d'interface à l'agent, lui permettant de percevoir et d'interagir dans l'environnement.

La perception est la capacité d'un agent à collecter des informations du système dans lequel il évolue. Généralement, la perception est vue comme un simple capteur récupérant des informations spécifiques. Pour notre part et en référence au domaine de la robotique [Begum et Karray, 2011, Elfes, 1989], nous définissons la perception d'un agent comme un processus complexe fondée sur trois étapes : (i) l'acquisition d'informations, (ii) leur extrapolation et (iii) leur filtrage.

L'acquisition a pour but d'acquérir les informations du système dans lequel évolue un agent. Vient ensuite l'extrapolation de l'ensemble des informations pour fournir plus de robustesse dans la collecte des informations. Finalement, des filtrages permettent à l'agent d'analyser les informations dont il a conscience et besoin. Autrement dit, par analogie à la perception humaine, un agent peut voir un objet sans le percevoir (en avoir conscience).

D'après [Jarras et Chaib-draa, 2002], les agents peuvent interagir en communiquant directement entre eux ou indirectement à travers l'environnement physique (la stigmergie).

La stigmergie est une méthode de communication indirecte où les individus communiquent entre eux en modifiant leur environnement et en percevant ces modifications. Nous pouvons citer comme exemple de ce type d'interaction celui des fourmis déposant des phéromones dans l'environnement. Une fourmi vagabonde dans l'environnement à la recherche de nourriture, puis dépose des phéromones tout au long du chemin lorsqu'elle en trouve. Toutes les fourmis sont attirées par ces phéromones en fonction de la quantité déposée. Ainsi, les fourmis interagissent par le biais de dépôts de phéromones dans l'environnement.

2.3/ VERS UN MODÈLE ORGANISATIONNEL POUR LES SYSTÈMES MULTIAGENTS HOLONIQUES

Les approches organisationnelles définissent la structuration des SMA, ainsi que les interactions possibles entre les agents en se focalisant sur un point de vue organisationnel.

D'après [Ferber et al., 2003], l'approche organisationnelle pour le domaine des systèmes multiagents offre les avantages suivant :
- La modularité : les organisations peuvent être vues comme des unités spécifiant le comportement de leurs membres. Ainsi, la conception du SMA est facilitée par la définition de règles pour chaque unité ;
- Les architectures multiples : l'approche organisationnelle ne restreint pas l'architecture interne de l'agent, ce qui permet de pouvoir implémenter tout type de modèle au niveau de l'agent ;
- L'hétérogénéité des langages : les groupes d'agents sont considérés comme des espaces d'interactions. A l'intérieur de ces groupes, les moyens spécifiques de communication peuvent être utilisés sans modifier l'architecture globale du SMA ;
- La sécurité des applications : tous les agents d'une organisation communiquent entre eux sans intervention du milieu extérieur. Cette caractéristique permet d'éviter les problèmes liés à la sécurité dans le SMA.

Dans le cadre de nos travaux, les deux premiers points nous poussent à utiliser l'approche organisationnelle. La complexité d'une foule nécessite de prendre en compte de nombreux aspects durant son analyse : le déplacement individuel, la constitution de groupes, les différentes interactions intra-groupes et inter-groupes spécifiques à chaque type de groupe (manifestants, policiers, promeneurs, *etc.*). De plus, l'environnement physique dans lequel sont immergés les piétons est lui même un système complexe nécessitant une analyse modulaire et pour lequel une approche organisationnelle de modélisation nous semble adaptée. La modularité et les architectures multiples deviennent essentielles pour mener à terme nos travaux et donner la possibilité d'une future réutilisation ou modification. Bien qu'importants, les deux derniers points exposés par [Ferber et al., 2003] ne sont pas considérés dans cette thèse.

Tout d'abord, nous décrivons les approches organisationnelles (dans la section 2.3.1), puis nous détaillons l'approche organisationnelle CRIO sur laquelle se base nos propositions (dans les sections 2.3.2, 2.3.3 et 2.3.4).

2.3.1/ LES APPROCHES ORGANISATIONNELLES

Les approches organisationnelles peuvent être classées comme suit : les hiérarchies, les holarchies, les coalitions, les équipes, les congrégations, les sociétés, les fédérations, les marchés et les organisations matricielles. Horling [Horling et Lesser, 2004] propose une étude de ces différents concepts, résumés ci-dessous.

Les hiérarchies [Mathieu et al., 2002, Montgomery et Durfee, 1993] organisent les agents dans une structure arborescente, où les agents plus élevés dans l'arbre ont une vision plus globale que leurs fils. Toutefois, les interactions entre entités sont indépendantes de la structure arborescente. Elles dépendent des connections existantes entre

les entités. Ce type de modèle est très efficace pour des systèmes hiérarchiquement décomposables en sous-systèmes. Ainsi, ces derniers peuvent utiliser un plus grand nombre d'agents [Yadgar et al., 2003].

Les holarchies [Rodríguez, 2005, Ulieru et Geras, 2002] sont dérivées du concept d'holon défini par [Koestler, 1967]. Il considère un système comme pouvant être défini soit d'un point de vue global : le système est vu *comme un tout*, soit d'un point de vue local : le système est un ensemble de parties interagissant entre elles. Comme pour les modèles hiérarchiques, les holarchies sont généralement utilisées dans des domaines où les objectifs peuvent être décomposés récursivement en sous-objectifs. Ce type d'approche permet la modélisation de systèmes complexes hiérarchiques, ainsi qu'une simulation multiniveau de ces systèmes. C'est sur cette approche que sont basés nos travaux. Nous reviendrons plus loin sur la description d'un modèle organisationnel holonique.

Les coalitions [Klusch et Gerber, 2002, Merida-Campos et Willmott, 2004] sont basées sur la résolution de problèmes par coopération d'agents, où ces derniers agissent dans leur propre intérêt tout en unissant leur savoir faire pour la résolution du problème. Ce type d'approche permet théoriquement un « *meilleur* » (par rapport à l'ensemble des objectifs des agents) ordonnancement des tâches [Shehory et Kraus, 1998].

Les équipes [Hexmoor et Beavers, 2001, Tambe, 1997] sont composées d'un certain nombre d'agents coopératifs qui ont convenu de travailler ensembles pour réaliser un but commun. En comparaison avec les coalitions, les équipes tentent de maximiser l'objectif global de l'équipe, plutôt que ceux des membres individuels. Les agents sont censés se coordonner de manière à ce que leurs actions individuelles soient compatibles avec et en faveur de l'objectif de l'équipe.

Les congrégations [Brooks et al., 2000, Griffiths, 2003] sont semblables aux coalitions et aux équipes. Toutefois, les organisations sont plates et forment des groupes d'individus. Les groupes d'individus sont formés suivant les similarités des caractéristiques des agents, où ces dernières sont définies suivant un ensemble de capacités ou d'exigences motivant la nécessité de se rassembler.

Les sociétés [Wellman et Wurman, 1998] sont basées sur le modèle de sociétés biologiques, où les agents sont définis suivant différents grades. Ces agents peuvent aller et venir, tandis que la société perdure. Cette société est alors vue comme un environnement dans lequel des agents se rencontrent et interagissent.

Les fédérations [Genesereth, 1997, Hayden et al., 1999] sont des modèles où les agents se regroupent et élisent un délégué pour les représenter, sacrifiant ainsi une part de leur autonomie [Genesereth, 1997, Wiederhold et al., 1992]. Les membres d'un groupe interagissent uniquement avec le délégué élu, qui agit comme un intermédiaire entre le groupe et le monde extérieur mais aussi entre les individus de son groupe.

Les marchés [Wellman, 2004] sont composés d'agents acheteurs et vendeurs. Les agents acheteurs peuvent demander ou faire des offres pour un ensemble d'éléments (comme le partage des ressources, des tâches, des services ou des biens). Les agents vendeurs sont responsables du traitement des offres et déterminent le gagnant d'une offre. Il s'agit d'une approche pour la modélisation de systèmes de type producteur-consommateur.

Les organisations matricielles [Wagner et Lesser, 2000] sont une généralisation des modèles hiérarchiques. À ces derniers est enlevée la contrainte d'une hiérarchie stricte et unilatérale, où un agent ne possède qu'un unique chef. Les agents peuvent ainsi recevoir, de différents agents, plusieurs objectifs à atteindre.

Aux vues des différentes approches proposées, les holarchies restent l'approche organisationnelle la plus adaptée à la conception de notre modèle. Notre modèle de l'environnement est alors vu comme un modèle hiérarchique où chaque niveau est une représentation plus précise que le niveau d'abstraction supérieur. Mais contrairement aux approches hiérarchiques, l'approche holonique permet de simuler le système (*i.e.* l'environnement) sur différents niveaux, où chacun d'eux est géré de manière autonome par un ensemble de holons. Ici, la société d'holon nous permettra de choisir dynamiquement le meilleur niveau à simuler suivant un ensemble de contraintes (*e.g.* la précision des résultats de simulations). Dans la section suivante, le métamodèle organisationnel CRIO est présenté. Ce métamodèle fournit l'ensemble des abstractions pour la modélisation de systèmes organisationnels et holoniques.

2.3.2/ LE MÉTAMODÈLE ORGANISATIONNEL CRIO

Nos travaux se basent sur le métamodèle organisationnel CRIO [Gaud et al., 2008a] destiné à la modélisation de systèmes complexes ouverts et de grande échelle. Un métamodèle se doit de définir de manière exhaustive l'ensemble des concepts manipulés dans le processus de développement. Le métamodèle CRIO est issu de l'intégration et l'extension de deux métamodèles existants. Le premier RIO a été proposé dans [Hilaire et al., 2000] et fut conçu pour la modélisation organisationnelle de SMA non hiérarchiques. Le second est le "*framework*" générique pour la modélisation de SMAH proposé dans [Rodríguez, 2005]. RIO tire son nom des trois principaux concepts sur lesquels il repose : Rôle, Interaction, et Organisation. CRIO, quant à lui précise et redéfinit certains des concepts précédemment introduits dans RIO et leur adjoint celui de Capacité. À cela, il intègre ensuite les concepts holoniques et s'intègre dans le processus de développement logiciel ASPECS qui sera décrit à la section 2.4.2.

CRIO s'inspire de l'approche de développement dirigée par les modèles [2] (ou MDD). Dans la logique de l'approche MDD, CRIO offre trois niveaux de modèles. Chacun d'eux est qualifié de domaine :

Le domaine du problème qui fournit la description organisationnelle du problème indépendamment d'une solution spécifique. Les concepts introduits dans ce domaine seront principalement utilisés durant la phase d'analyse et au début de la phase de conception du processus de développement.

2. MDD : "*Model Driven Development*"

Le domaine agent qui introduit les concepts multiagents et fournit une description d'une solution multiagent, éventuellement holonique, basée sur les éléments du *domaine du problème*. Le *domaine agent* est davantage associé à la fin de la phase de conception.

Le domaine de la solution est relatif à l'implantation de la solution sur une plateforme spécifique. Cet aspect est donc dépendant d'une plateforme de déploiement particulière.

Nous avons choisi CRIO, car ses concepts permettent de modéliser un système complexe suivant l'approche organisationnelle. De plus, ce métamodèle fournit les outils pour réaliser une projection du modèle organisationnel sur une société d'agents, puis son implantation, réduisant ainsi les problèmes liés entre le passage des différentes étapes du processus de conception de notre modèle.

Les concepts définis dans le *domaine du problème* et dans le *domaine agent* seront décrits dans la suite de ce chapitre (*cf.* sections 2.3.3 et 2.3.4). Tout au long de ce chapitre, la notation UML est utilisée pour décrire chacun de ces métamodèles. Différents *"profiles"* UML sont également introduits afin d'adapter les diverses notations UML aux besoins spécifiques de l'approche proposée.

2.3.3/ LE DOMAINE DU PROBLÈME DU MÉTAMODÈLE CRIO

La figure 2.1 présente le diagramme UML de la partie du métamodèle CRIO [3] consacrée à la modélisation d'un problème [4]. Le domaine du problème introduit les concepts au cœur de l'approche proposée : organisation, rôle, interaction, capacité et ontologie. L'ensemble de ces concepts est détaillé dans la suite de cette section.

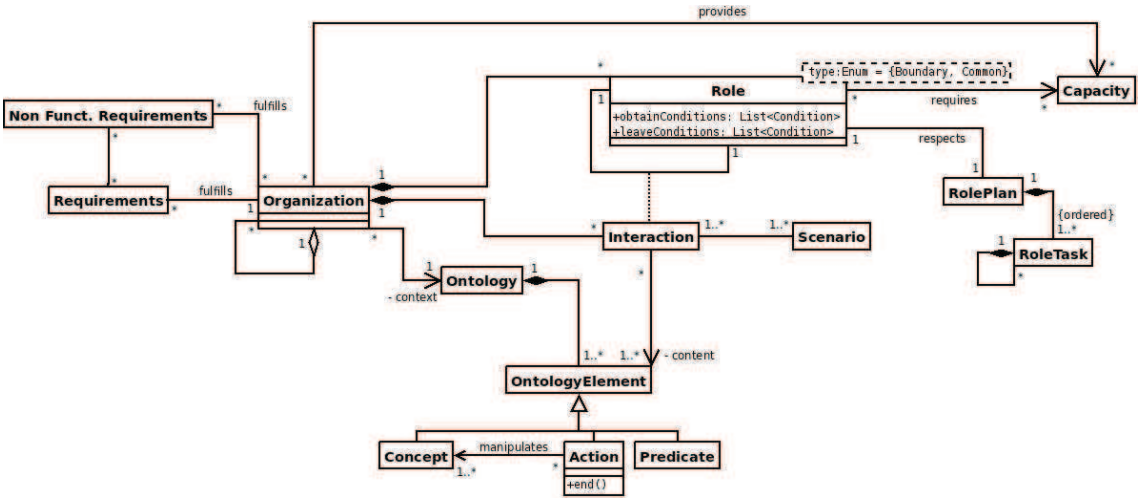


FIGURE 2.1 – Diagramme UML du domaine du problème du métamodèle CRIO

3. http://www.aspecs.org/Problem_Domain
4. Nous rappelons qu'une classe liée à une association par des pointillés correspond à une classe d'association (*e.g.* : Interaction). Il s'agit d'une classe qui réalise la navigation entre les instances d'autres classes.

2.3.3.1/ ONTOLOGIE

L'ontologie (*"Ontology"*) et les éléments d'ontologie (*"Ontology Element"*) sont utilisés pour décrire les connaissances du domaine de l'application et définir le contexte des organisations du système. Une ontologie est un ensemble structuré de concepts visant à décrire un ou plusieurs domaines d'étude. Les relations entre concepts peuvent être sémantiques, de composition ou d'héritage.

[Gruber, 1995] fournit une définition générale de la notion d'ontologie : *« Une ontologie est la spécification d'une conceptualisation d'un domaine de connaissance »* . Dans [Gaud, 2007], une définition plus opérationnelle, proposée par l'OMG dans le document de spécification *"Ontology Definition Metamodel"*[5] (ODM), est adoptée : *« Une ontologie définit un ensemble de termes et de concepts communs utilisés pour décrire et représenter un domaine de connaissance. Une ontologie peut prendre diverses formes telles qu'une taxonomie (ensemble de connaissances avec une hiérarchie minimale), un thésaurus (mots et synonymes), un modèle conceptuel (avec des connaissances plus complexes) ou une théorie logique (avec des connaissances très riches, complexes, consistantes et significatives). Une ontologie bien formée est celle qui est exprimée dans une syntaxe bien définie et qui dispose d'un interpréteur bien précis et conforme avec la définition ci-dessus. »*

Dans le métamodèle CRIO, l'ontologie possède une double fonction. Elle permet tout d'abord de rassembler et d'organiser l'ensemble des connaissances disponibles sur le problème et sur son domaine, et de définir le contexte des organisations utilisées pour les modéliser. La structuration des concepts de l'ontologie du problème permet notamment de renseigner le concepteur sur la structure éventuelle du système. Concernant l'aspect de capitalisation des connaissances du domaine, l'ontologie constitue également une base de connaissances commune à tous les acteurs intervenant dans le développement du système. Elle permet ainsi de regrouper les concepts issus du vocabulaire spécifique des experts du domaine et de les organiser de sorte à faciliter leur compréhension par les équipes en charge de la conception du système.

L'ontologie regroupe également l'ensemble des connaissances qui seront échangées au cours des interactions entre les rôles qui composent les organisations du système. L'ontologie constituera ainsi dans le domaine agent la base de connaissances nécessaire à la définition des communications entre les agents.

Une ontologie est composée d'*éléments d'ontologie* (abstraits) disposant de trois types concrets possibles (*cf.* figure 2.1) :

Concept : une catégorie, une abstraction qui abrège et résume une multiplicité d'objets par généralisation de traits communs identifiables.

Action : mécanisme réalisé par un acteur qui modifie une ou plusieurs propriétés (et par conséquent leurs états) d'un ou plusieurs concepts récepteurs.

Prédicat : assertions sur les propriétés des concepts.

2.3.3.2/ ORGANISATION

L'organisation est, avec l'interaction, l'un des concepts clefs des approches organisationnelles dans les SMA. D'ailleurs ces deux concepts sont intrinsèquement liés. L'interaction

5. http://www.omg.org/ontology/

a lieu au sein d'une organisation, mais réciproquement l'organisation exige l'existence des interactions puisqu'elle-même est définie sur leur base. Selon [Ferber, 1995], les organisations constituent à la fois le support et la manière dont se déroulent les interactions, c'est-à-dire la façon dont sont réparties les tâches, les informations, les ressources, et la coordination des actions. La vision proposée de l'organisation, présentée dans la définition 2.10, est très proche de ce que [Ferber, 1995, chap. 3] qualifie de *structure organisationnelle*.

[Gaud, 2007] donne la définition du concept d'organisation (*cf.* définition 2.10), incluant les concepts de rôle et d'interaction, concepts retrouvés dans les approches organisationnelles. L'objectif d'une organisation est de satisfaire à des besoins. Chaque besoin doit être satisfait par un unique rôle ou un ensemble des rôles composant l'organisation. Cette dernière est considérée soit comme la description d'un comportement global d'un ensemble d'entités, soit comme un ensemble de comportements en interaction. La dualité que procure une organisation permet donc de définir des hiérarchies organisationnelles en décomposant la partie d'un rôle en une organisation.

Définition 2.10 Organisation [Gaud, 2007], inspiré de [Hilaire et al., 2000, Rodríguez, 2005]

Une organisation est définie par un ensemble de rôles, leurs interactions et un contexte commun définissant ainsi un schéma d'interaction spécifique. Le contexte d'une organisation est défini par tout ou partie de l'ontologie du domaine associée aux besoins dont elle est en charge.

Chaque organisation est associée à au moins un besoin fonctionnel qui correspond soit à l'objectif qu'elle doit satisfaire, à la tâche qu'elle doit effectuer ou au comportement global qu'elle doit exhiber. L'objectif de chaque organisation est donc de satisfaire les différents besoins fonctionnels et non-fonctionnels auxquels elle est associée. Chacun d'entre eux devra être satisfait soit par le comportement individuel de l'un des rôles de l'organisation, soit par le comportement global émergeant des interactions de tout ou partie des rôles.

L'organisation étant à la fois le support et la manière de satisfaire un besoin, elle peut être considérée comme la description d'un comportement global auquel devront se plier une ou plusieurs entités, afin de satisfaire les objectifs qui leurs sont attribués. Si l'on cherche à étudier ce comportement et à le décomposer d'un point de vue fonctionnel, on obtient un ensemble de comportements de complexité inférieure interagissant pour satisfaire les objectifs associés à l'organisation. Selon le niveau d'abstraction considéré, une organisation peut être vue soit comme un comportement unitaire soit comme un ensemble de comportements en interaction. L'organisation est donc un concept intrinsèquement récursif.

L'organisation est à la fois l'agrégation d'un ensemble de comportements, et un comportement composant une organisation de niveau supérieur ; le tout constituant une hiérarchie de comportements spécifiques avec à chaque niveau des objectifs précis à satisfaire. Cette définition récursive de l'organisation constituera la base du processus d'analyse associé à ASPECS. Les différents comportements qui composent un système seront récursivement décomposés en un ensemble de sous-comportements en interaction. Cette décomposition se poursuit jusqu'à atteindre un niveau où des comportements obtenus

sont considérés comme suffisamment simples pour être gérés par des entités atomiques faciles à implanter.

2.3.3.3/ RÔLE

Les comportements considérés comme élémentaires à un niveau d'abstraction donné sont appelés *rôles*. La définition 2.11 a été retenue pour définir le concept de *rôle*. Cette vision de l'organisation qui est perçue tantôt comme comportement à part entière, tantôt comme un rôle dans une organisation de niveau d'abstraction supérieur est partagée par d'autres auteurs : [Anderson et Reenskaug, 1992] et [Singh, 1992].

Chaque rôle est défini sur une et une seule organisation. Dans le contexte de son organisation, un rôle définit un comportement et un statut, ces deux composantes du rôle sont détaillées ci-après.

Définition 2.11 Rôle [Gaud, 2007], inspiré de [Hilaire et al., 2000, Rodríguez, 2005]

Un rôle est l'abstraction d'un comportement dans le contexte (espace d'interaction) d'une organisation. Il confère à l'entité qui le joue un statut dans cette organisation. Ce statut définit un ensemble de droits et de devoirs. Un rôle peut interagir avec les autres rôles définis par la même organisation.

Le statut d'un rôle définit la position de ce dernier au sein de son organisation ainsi qu'un ensemble de droits et d'obligations pour l'agent qui le joue. Le statut définit également l'interface au travers de laquelle l'agent jouant le rôle est perçu par les autres entités de la même organisation. Il fournit à cet agent le droit d'exercer ses capacités (compétences) dans le contexte de l'organisation et l'obligation de respecter le comportement décrit par le rôle. Le statut d'un rôle est caractérisé par au moins un concept de l'ontologie définissant le contexte de son organisation.

Le comportement d'un rôle fixe les responsabilités qui sont associées au rôle et la méthode pour les satisfaire. Pour définir ce comportement, chaque rôle dispose d'attributs qui lui sont propres. L'objectif d'un rôle est de contribuer pour toute ou partie aux besoins associés à l'organisation dans laquelle il est défini. Le comportement d'un rôle est spécifié par un plan comportemental (ou plan de comportement) qui peut être représenté par un diagramme d'activité UML ou un diagramme état-transition ("*state-chart*"). Un tel plan décrit comment un ou plusieurs objectifs d'un rôle peuvent être satisfaits par son comportement. Il détaille comment combiner et ordonner les interactions avec les autres rôles, les événements extérieurs au rôle, et avec les tâches qui composent le comportement du rôle.

L'aspect dual du rôle dans CRIO, combinant statut et comportement, est une différence majeure avec bien des définitions existantes de ce concept. En effet, le rôle est souvent considéré comme une interface au travers de laquelle les autres agents perçoivent l'agent qui le joue. Le rôle est alors assimilé à une sorte de filtre pour l'agent, ce qui correspond à la notion de statut dans l'approche proposée. Dans MESSAGE [Caire et al., 2002] par exemple, la relation entre le rôle et l'agent est considérée comme analogue à celle entre l'*interface* et la *classe* dans les modèles orientés-objets. [Kristensen et Osterbye, 1996],

étudiant la notion de rôle pour les objets, nomment cette approche la métaphore du filtre (ou *"Filter Metaphor"*) et examine les problèmes liés à une telle vision : *"This is mistaken because the filter metaphor implies that the persons*[6] *has all the properties from the outset, and we choose to see only some of them. This neglects the important meaning behind roles, that the properties are extrinsic, — the person only has them because of the role"*. Une étude du concept de rôle pour les objets, qui partage d'ailleurs de nombreux points communs avec l'approche présentée ici consacrée aux rôles d'agents, peut-être trouvée dans la thèse de [Graversen, 2006].

Après cette description des aspects internes à un rôle, la suite immédiate est logiquement consacrée à l'étude des relations entre rôles. La base de la coordination entre rôles est incarnée par l'interaction.

2.3.3.4/ INTERACTION

De nombreux auteurs s'accordent sur le fait que la complexité des SMA est une conséquence directe de l'interaction entre les agents [Jennings, 2001, Odell, 2002]. La notion d'interaction est fondamentale puisqu'elle permet à un groupe d'agents d'accomplir davantage ensemble que la somme de leurs actions individuelles. Mais l'interaction va également de pair avec la nécessité de coordination et l'émergence de conflits. Une interaction entre rôles est définie de la manière suivante :

Définition 2.12 Interaction [Gaud, 2007], inspiré de [Hilaire et al., 2000, Rodríguez, 2005]

Une interaction entre rôles est composée de l'événement généré par un premier rôle et perçu par les autres, ainsi que les réactions induites dans ces derniers. Les rôles qui interagissent partagent automatiquement un contexte d'interaction commun.

Les interactions survenant entre les rôles au sein d'une organisation sont décrites dans un scénario d'interaction qui est généralement représenté par un diagramme de séquence UML. Un scénario d'interaction décrit comment un ensemble de rôles interagissent et se coordonnent pour satisfaire un objectif commun, lequel peut lui même être associé à un rôle de niveau d'abstraction supérieur. Cette association implique de pouvoir faire transiter de l'information entre deux niveaux d'abstraction adjacents. Ce mécanisme est géré dans CRIO par l'utilisation combinée des concepts de capacité, d'organisation et de service. Ce dernier sera présenté à la section 2.3.4.4 de ce chapitre.

Le concept d'interaction du modèle CRIO diffère de celui présenté dans les SMA, car il s'agit d'interactions entre rôles et non plus entre agents. Les interactions peuvent servir soit à faire transiter simplement de l'information, soit à coordonner les tâches de différents rôles pour satisfaire un objectif commun. Elles sont généralement normalisées par les protocoles de communications inter-agents comme [FIPA, 1998].

6. Le terme *"Personne"* est à comprendre ici dans le sens de l'entité qui joue le rôle

2.3.3.5/ CAPACITÉ

Les systèmes de grande échelle doivent pouvoir coopérer et fonctionner en environnement ouvert. Les agents impliqués dans de tels systèmes doivent par conséquent collaborer avec d'autres agents, éventuellement intégrés à des systèmes différents, pour satisfaire leurs objectifs. Il en découle que tout agent doit être en mesure d'évaluer les compétences de ses partenaires potentiels et ainsi identifier les collaborateurs les plus appropriés. La notion de capacité fut initialement introduite pour permettre aux agents de raisonner sur leurs propres compétences et celles de leurs collaborateurs de sorte à pouvoir s'adapter et satisfaire des objectifs nouveaux [Rodríguez et al., 2007].

L'objectif de l'introduction du concept de capacité est de disposer d'un outil permettant de décrire les compétences d'un agent ou d'un groupe d'agents de manière générique, tout en faisant abstraction de l'architecture interne de l'agent (*cf.* définition 2.13).

Définition 2.13 Capacité [Rodríguez et al., 2007]

Une capacité est la description abstraite d'un savoir-faire. Elle regroupe la description de moyens qui permettent l'accomplissement d'une tâche. Ceci incarne une fonctionnalité logique aux yeux des entités qui la fournissent (les propriétaires) et de celles qui l'utilisent ou la requièrent (les utilisateurs).

Les propriétaires de capacités peuvent être des agents, des rôles ou des organisations. Les utilisateurs sont généralement des rôles.

Ainsi, la capacité possède une double fonction dans le métamodèle CRIO. (i) La capacité permet tout d'abord, dans le processus de modélisation, d'effectuer l'interface entre deux niveaux d'abstraction adjacents dans la hiérarchie organisationnelle du système. (ii) Elle constitue également une interface entre l'agent et le rôle, permettant de définir le comportement du rôle en faisant abstraction de l'architecture de l'agent. Cette notion permet d'obtenir des modèles génériques d'organisation. Une capacité représente en effet une compétence d'un agent ou d'un groupe d'agent. La capacité permet d'effectuer l'interface entre le rôle et l'entité qui le joue. Le rôle requiert certaines compétences pour définir son comportement, lesquelles sont modélisées par des capacités. Les capacités peuvent ensuite être invoquées dans l'une des tâches composant le comportement du rôle. Une entité souhaitant jouer un rôle devra fournir une réalisation concrète (ou implantation) à chacune des capacités requises par le rôle.

2.3.4/ LE DOMAINE AGENT DU MÉTAMODÈLE CRIO

Le *domaine du problème* de CRIO permet la modélisation du problème en termes d'organisations, de rôles, de capacités et d'interactions. Le résultat de cette modélisation doit aboutir à la définition d'une hiérarchie d'organisations combinant leurs comportements respectifs pour satisfaire les besoins identifiés. Disposant désormais d'un modèle du problème, l'objectif est alors de concevoir le *modèle d'une solution multiagent*. Il s'agit d'élaborer le modèle d'une société d'agents capables d'offrir une solution au problème étudié, en décrivant les interactions entre agents ainsi que leurs éventuelles dépendances. La

figure 2.2 présente le diagramme UML du *domaine agent* du métamodèle CRIO [7].

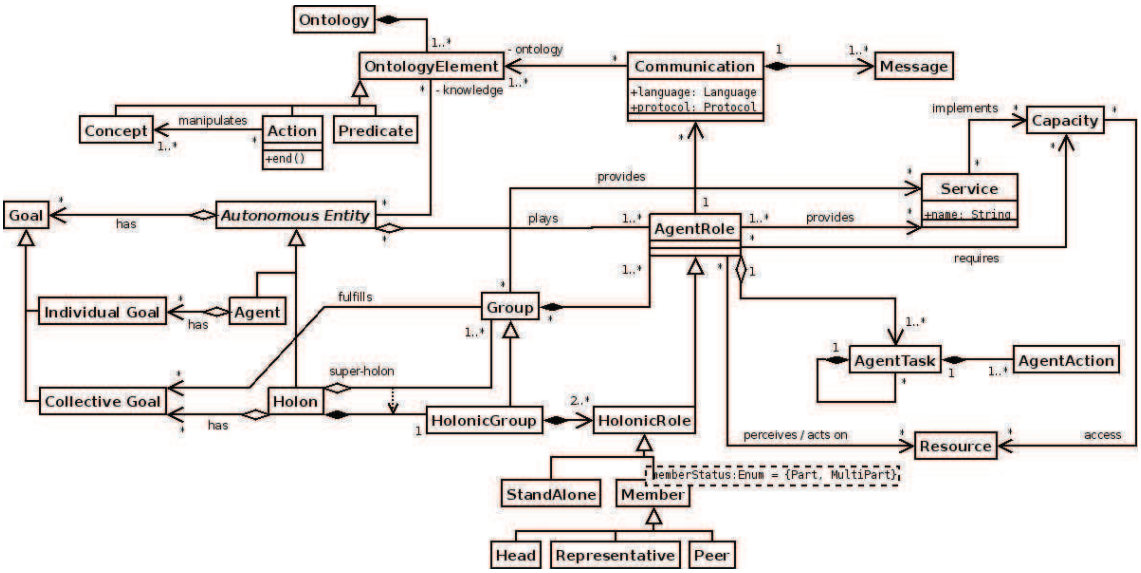


FIGURE 2.2 – Diagramme UML du domaine de l'agent du métamodèle CRIO

2.3.4.1/ DE L'ORGANISATION AU GROUPE

Au sein du *domaine agent*, les organisations issues du modèle du problème sont instanciées sous forme de *groupes*. Les rôles qui composent ces organisations sont eux-aussi instanciés sous forme de *rôles d'agent* (ou *"AgentRole"*). Ces concepts de *groupe* et de *rôle d'agent* sont définis dans les définitions 2.14 et 2.15.

Définition 2.14 Groupe [Gaud, 2007]

Un groupe est une instance d'une organisation. Il modélise un groupe d'agents en interaction, coopérant pour satisfaire un ou plusieurs objectifs.

Deux agents ne peuvent communiquer que si ils jouent un rôle dans un groupe commun. Un agent, lorsqu'il joue un rôle dans un groupe donné, se doit de respecter le comportement de ce rôle. Le comportement global du groupe doit suivre le schéma spécifique d'interaction décrit par l'organisation qu'il instancie.

Définition 2.15 Rôle d'agent [Gaud, 2007]

Un rôle d'agent est une instance concrète d'un rôle. Il décrit un comportement dans le contexte défini par un groupe. Il confère à l'agent un statut dans ce groupe et les moyens d'interagir avec les autres agents jouant des rôles au sein du même groupe.

7. http://www.aspecs.org/Agency_Domain

Un rôle d'agent est local à un groupe. L'accès à un tel rôle n'est pas automatique : il doit être demandé par l'agent qui désire le jouer. Cet accès est soumis à des conditions que doit valider l'agent pour accéder effectivement au rôle. Les conditions d'obtention d'un rôle sont nommées `obtainConditions`, et l'une d'elles spécifie que l'agent doit disposer de toutes les capacités requises par le rôle. Ces conditions d'obtention constituent un moyen de vérifier que l'agent valide effectivement un certain nombre de pré-requis nécessaires au rôle et notamment qu'il dispose de toutes les compétences requises pour mettre en œuvre le comportement du rôle. Un rôle ne peut pas être libéré directement, car sa libération est également soumise à des conditions. Les conditions de libération, nommées `leaveConditions`, permettent par exemple d'empêcher un agent de quitter un groupe alors qu'il avait la charge d'effectuer une tâche donnée.

Ces différentes conditions permettent de fixer le niveau d'engagement d'un agent dans un groupe. En somme, lorsqu'un agent entre dans un groupe, il accepte non seulement de se plier aux comportements définis dans les rôles qu'il joue, mais également de mettre à disposition tout ou partie de ses compétences. Ces aspects sont directement liés au statut du rôle qui définit les droits dont dispose l'agent, mais également les obligations auxquelles il doit se plier.

Un agent doit jouer au moins un rôle dans un groupe pour pouvoir communiquer, mais il peut également jouer plusieurs rôles au sein d'un même groupe ou plusieurs rôles au sein de groupes différents.

Au cours du processus d'instanciation, une même organisation peut être instanciée plusieurs fois. De manière analogue un rôle peut être instancié sous forme de plusieurs rôles d'agent, plusieurs fois au sein d'un même groupe ou au sein de groupes différents. La relation entre les concepts d'organisation et de groupe, et celle entre rôle et rôle d'agent peuvent être considérées comme analogues à la relation qui lie les concepts de *Classe* et d'*Objet* dans les métamodèles orientés-objets. Les définitions et les relations entre les concepts d'agent, de groupe et de rôle d'agent au sein du *domaine agent* sont très proches de celles fournies dans le métamodèle Agent/Groupe/Rôle (AGR) [Ferber et al., 2004].

2.3.4.2/ De l'interaction à la communication

Le modèle de communication utilisé entre les rôles d'agent présuppose que deux rôles, qui interagissent, partagent au moins le savoir lié au contexte de leur groupe, mais également celui spécifiquement lié à l'information échangée au cours de l'interaction.

Une communication est une spécialisation d'une interaction dans laquelle les connaissances partagées par les participants sont représentées par un ensemble d'éléments de l'ontologie. Une conversation est une spécialisation d'une communication dans laquelle le contenu (langage, ontologie, et encodage) et le contrôle de la communication (protocole) sont explicitement détaillés. Une conversation est principalement composée d'actes de langage [FIPA ACL, 2002, FIPA Com. Act, 2002]. Un protocole définit une séquence de messages attendus et représente un schéma commun de conversation utilisé pour exécuter une tâche, une stratégie de haut niveau gouvernant l'échange d'information entre les agents. Chacun des participants connaissant ce schéma, leur dialogue en est ainsi facilité.

2.3.4.3/ AGENT ET HOLON

Le cœur du *domaine agent* tient évidemment dans la définition des entités qui seront à la base de la solution du problème traité : les agents.

Définition 2.16 Agent [Gaud, 2007]

Un agent est une entité jouant un ensemble de rôles qui peuvent être définis dans plusieurs groupes. L'agent est caractérisé par ses rôles, ses connaissances et ses capacités.

Derrière cette définition, nous retrouvons un concept relativement proche de la définition fournie par [Ferber, 1995] (*cf.* section 2.2.1). Cependant, des parties du comportement d'un agent peuvent être désormais externalisées dans ses rôles ou ses capacités. Comparé à la définition de Ferber, cette dernière définie un agent d'un point de vue abstrait. Toutefois, les notions d'interaction, de perception et d'autonomie sont implicites. De plus, cette définition implique la coexistence de rôles que jouerais les agents, alors que pour Ferber le rôle définit le statut des agents.

En effet, chaque agent dispose de la capacité de base qui lui permet de jouer des rôles et donc de communiquer avec d'autres agents. Chaque agent possède ensuite des capacités ou compétences spécifiques, qu'il peut mettre au service d'autres agents, soit en partageant un groupe commun, soit par la publication d'un service (*cf.* section 2.3.4.4).

Le comportement global d'un agent résulte de la combinaison des rôles, connaissances et capacités dont il dispose à un instant donné. Chaque agent tend à satisfaire les objectifs qui ont été attribués aux rôles qu'il joue, et qui sont spécifiés dans leur comportement. Mais le comportement d'un rôle peut indirectement être influencé par les caractéristiques propres de l'agent telles ses accointances ou ses tendances, si l'implantation d'une des capacités qu'il requiert, est basée sur ces éléments. L'accès aux caractéristiques propres de l'agent ne s'effectue généralement que par l'intermédiaire d'une implantation particulière d'une capacité requise par le rôle. Cette approche garantit un certain niveau de généricité du rôle en faisant abstraction de l'architecture interne de l'agent.

Un aspect important à retenir dans la définition 2.16 est qu'elle autorise, voire favorise, une dynamique importante des rôles au sein de l'agent, et du système dans sa globalité. Un agent jouera simultanément et/ou successivement au cours de son exécution un grand nombre de rôles.

En sus de la notion d'agent, CRIO introduit un second type d'entité incarné par la notion de holon. Dans cette thèse, le terme holon désigne des agents composés d'agents. Dans la littérature, il existe d'autre termes permettant de définir ce concept comme : les *groupes agentifiés* [Odell et al., 2005], les *agents récursifs* ou *intermédiaires* [Correa e Silva Fernandes, 2001, Holland, 1995], les *méta-agents* [Holland, 1995] ou les *agents collectifs* et *individuels* [Ferber, 1995].

Ainsi, les holons peuvent être utilisés pour la modélisation de systèmes complexes en introduisant plusieurs niveaux de représentation. Pour illustrer cette notion de niveau de représentation, prenons l'exemple de la modélisation d'une université, représentée par la figure 2.3.

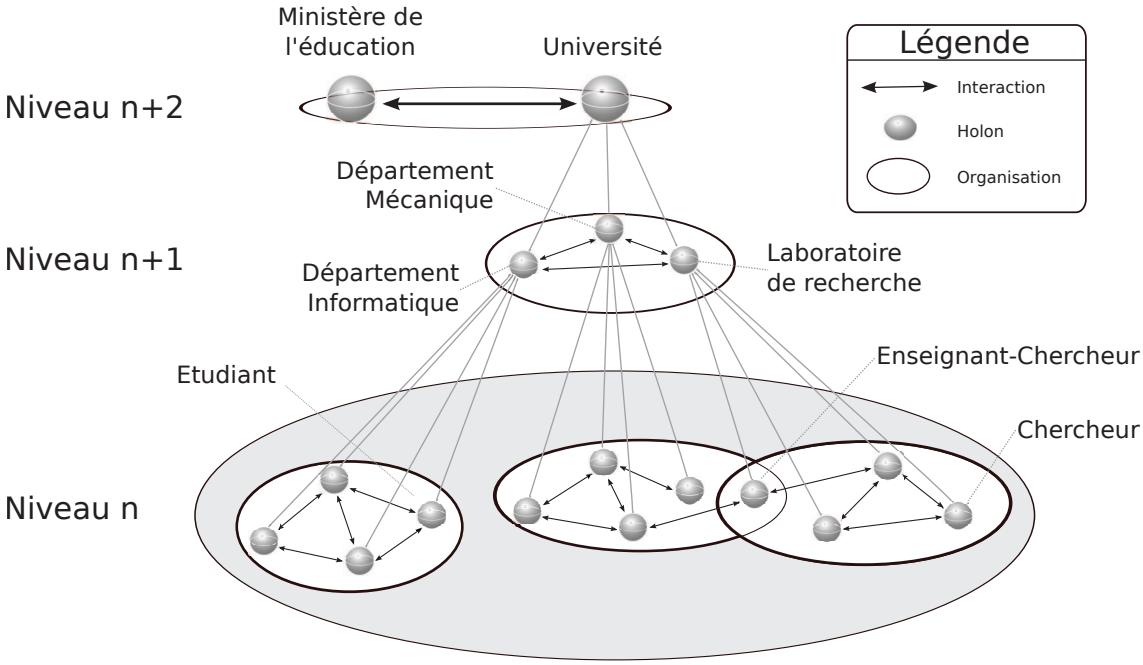


FIGURE 2.3 – Exemple d'une représentation holonique d'une université [Gaud, 2007]

Conceptuellement, une organisation d'un niveau n fournit un raffinement d'un holon de niveau strictement supérieur à n. Nous pouvons ainsi dire qu'un laboratoire de recherche contribue à une université, car le laboratoire est un sous-système de ce dernier. Par rapport aux SMA classiques, de nouvelles interactions voient le jour. Il s'agit des interactions intra-holon pour permettre l'échange d'informations entre deux niveaux d'un holon. Elles font le lien entre un système et ses sous-systèmes. Toutefois, notons que ces interactions sont propres à l'implémentation d'un holon. Pour la modélisation d'un SMAH, il est donc nécessaire d'utiliser des notions pouvant faire le lien entre les différents niveaux d'abstractions d'une hiérarchie. Dans le métamodèle CRIO, ce lien est supporté par les concepts de capacité et de service [Gaud, 2007, Rodríguez et al., 2007, Rodríguez, 2005], présentés dans la section suivante.

2.3.4.4/ RELATIONS ENTRE LES CONCEPTS DE CAPACITÉ ET DE SERVICE

Il était nécessaire, pour aborder sereinement la partie qui suit, de comprendre les fondements de la vision proposée pour la notion de holon ainsi que la manière adoptée pour structurer les membres d'un holon composé. Il est maintenant possible de s'intéresser aux relations qui existent entre les concepts de service et de capacité pour montrer comment un super-holon peut pleinement exploiter les compétences de ses membres pour accéder à des rôles qui leur sont inaccessibles. La section présente se focalise en effet sur l'exploitation des compétences émergentes d'un groupe de holons en interaction. La clef d'un tel mécanisme repose sur le fait qu'un groupe d'agent est en mesure de fournir un service. Dans l'approche proposée, la notion de service se définit de la manière suivante :

Définition 2.17 Service [Gaud, 2007]

Un service est une fonctionnalité qui représente la possibilité d'exécuter des tâches et ce au nom de son propriétaire : agent, holon, rôle ou groupe. Cette fonctionnalité logique est généralement associée à une ou plusieurs capacités qu'elle est capable de réaliser.

D'après la définition précédente, un agent ou un groupe est en mesure de fournir un service qui peut réaliser une capacité. Si on prend le cas d'un rôle, cette définition prend un sens tout particulier. En effet, un rôle requiert des capacités pour définir son comportement, mais il peut également les mettre à la disposition d'autres rôles en publiant un service. Les différents rôles d'un groupe peuvent alors exploiter leurs capacités respectives par échanges de services. Un groupe est également capable de fournir une capacité *« collective »* , qu'il peut partager avec les autres groupes en publiant un service à son tour. Ces capacités de groupe résultent de la collaboration entre les différents membres du groupe.

Cette section montre comment exploiter dans le cadre des SMAH, ces compétences qui *émergent* des interactions entre les différentes parties d'un système. En effet, le groupe est considéré comme capables de fournir un service. Or, un service peut être vu comme une manière possible de réaliser une capacité. Dès lors, un super-holon peut posséder une capacité, qui peut être réalisée par un service, fourni par tout ou partie de ses membres.

Un groupe peut réaliser une capacité via un service. Ce type de réalisation implique que deux niveaux d'abstraction doivent cohabiter au sein d'un même système. L'approche holonique permet de le modéliser de manière aisée.

En effet, un super-holon peut exploiter les comportements émergents des interactions entre ses membres et ainsi jouer des rôles qui ne pourraient être joués par les membres du super-holon, car ces rôles sont définis dans une autre organisation où est situé le super-holon. Un super-holon peut donc réaliser une capacité, soit en obtenant directement une réalisation sous forme d'algorithmes, soit en intégrant un groupe capable de fournir un service qui réalise la capacité dont il a besoin.

Une organisation peut fournir un service de différentes manières. On peut distinguer trois approches possibles :

Atomique : le service est fourni par l'un des rôles de l'organisation. Le rôle possède une capacité disposant d'une description compatible avec celle du service.

Composé : le service est obtenu depuis les interactions d'un sous-ensemble des rôles de l'organisation en suivant un protocole connu. Ceci est un scénario de composition de service où le plan d'obtention est a priori connu et où les performances peuvent être connues et assurées.

Émergent : le service est obtenu grâce aux interactions d'un sous-ensemble de rôles de l'organisation, mais le plan d'obtention n'est pas connu a priori. Le résultat du service est émergent.

Dans ce dernier cas, la gestion du service fourni par le comportement global de l'organisation est généralement assurée par l'un des rôles de l'organisation.

Dans les sections précédentes, le métamodèle CRIO a été introduit pour faciliter l'analyse et la conception de systèmes complexes. Celui-ci fournit trois métamodèles ; chacun étant associés à l'un des niveaux de modélisation de l'approche de développement dirigée par modèles (MDD) : le modèle métier indépendant de l'informatisation (« *Computation Independent Model* » , CIM), le modèle indépendant de la plateforme (« *Platform Independent Model* » , PIM), et le modèle spécifique à la plateforme cible (« *Platform Specific Model* » , PSM) [MDA, 2003]. Seuls les deux premiers niveaux ont été détaillés dans ce chapitre. La conception de notre modèle pour la simulation de foule requiert, en complémentarité de CRIO, une plateforme agent (*cf.* section 2.5). De plus, nous adopterons une démarche rigoureuse durant le processus de conception de notre architecture en utilisant la méthodologie « *orientée-agents* » présentée dans la section suivante. Nous présenterons à tour de rôle la méthodologie ASPECS (basé sur le métamodèle CRIO) pour la conception de notre modèle et la plateforme JANUS (proposant une implantation du métamodèle CRIO).

2.4/ MÉTHODOLOGIES ORIENTÉES-AGENTS

Une simulation repose sur l'exécution d'un modèle. Pour faciliter sa conception et sa description, nous nous basons sur une méthodologie. Une méthodologie fournit un processus rigoureux allant d'une étape analysant les besoins du problème, en passant par la mise en place d'une solution et la création d'un prototype, jusqu'à celle concernant le déploiement du prototype. La sous-section qui suit présente rapidement les principes des méthodologies orientées-agents. Une seconde sous-section détaille la méthodologie ASPECS que nous utiliserons pour concevoir notre modèle dans les chapitres 4 et 5, ainsi que dans l'annexe A.

2.4.1/ PRINCIPES SUR LES MÉTHODOLOGIES

Cette sous-section présente les principes des méthodologies orientées-agents. La définition d'une méthodologie non orientée-agents est donnée par [Ghezzi et al., 2002] (*cf.* définition 2.18). La conception de notre modèle ayant été fait à l'aide de la méthodologie ASPECS, il est essentiel de rappeler les différents aspects d'une méthodologie. Cette partie s'inspire de la synthèse faite par [Gaud, 2007] sur les principes des méthodologies.

Définition 2.18 Méthodologie [Ghezzi et al., 2002]

Une méthodologie est une collection de méthodes couvrant et reliant les différentes étapes d'un processus. L'objectif d'une méthodologie est de prescrire une approche cohérente à adopter pour résoudre un problème dans le contexte d'un processus de développement de logiciel en pré-sélectionnant et reliant un certain nombre de méthodes.

La plupart des méthodologies existantes s'inspirent des résultats et des contributions issus de l'ingénierie logicielle orientée-objets en y intégrant les spécificités liées à l'ap-

proche agent telles que l'autonomie ou la nature sociale des agents (PASSI [Chella et al., 2006, Cossentino, 2005], MESSAGE [Caire et al., 2002], AAII [Kinny et al., 1996], MaSE [DeLoach et al., 2001], ADELFE [Bernon et al., 2002]). Ce constat n'a d'ailleurs rien d'étonnant vu les nombreux points communs que partagent les concepts d'agent et d'objet.

Dans les quinze dernières années, de nombreux efforts de l'ingénierie logicielle orientée-agents se sont portés sur la définition de méthodologies pour guider le processus de développement des SMA. La conception de la plupart des méthodologies (orientées-agents ou non) débute généralement par la définition d'un métamodèle qui identifie les abstractions de base nécessaires à la description du problème et de sa solution. Ces différentes abstractions sont ensuite organisées afin de définir les étapes à suivre pour l'analyse, la conception, et l'implantation ainsi que les résultats que chacune d'elles doit produire. Malheureusement, ceci est loin d'être suffisant du point de vue de la mise en pratique du développement logiciel [Cernuzzi et al., 2005]. En effet, dans le domaine du développement de systèmes logiciels et donc des SMA, l'identification d'une méthodologie appropriée passe d'abord par l'identification d'un modèle spécifique pour son processus de développement logiciel [Boehm, 1988] (*cf.* définition 2.19).

Définition 2.19 Processus de développement logiciel [Fuggetta, 2000]

Un processus de développement de logiciel est l'ensemble cohérent des politiques, structures organisationnelles, technologies, procédures, et diagrammes nécessaires à la conception, au développement, au déploiement et à la maintenance (évolution) d'un produit logiciel.

Un tel modèle doit définir : (i) l'organisation et la coordination des différentes phases du développement ; (ii) les différentes personnes qui doivent intervenir dans chaque activité et à quel moment ; (iii) les technologies et les outils qui doivent être utilisés dans chaque activité ; (iv) les produits fournis par chaque activité ; et (v) les ressources impliquées dans chaque phase du processus de production. En d'autres termes, le modèle de processus et le processus qui en découle doivent guider l'ensemble de l'effort de production et être complétés par un ensemble de lignes méthodologiques spécifiques à la méthodologie développée. Le terme *« méthodologie »* est donc souvent employé dans un sens qui le réduit à une sous-partie de son processus et à ses produits. Cette partie n'abordera pas les différents types de modèles de processus de développement (Cascade, Spirale, Fontaine, *etc.*). Cependant, d'après [Sommerville, 2004], les différents types de processus de développement logiciel partagent quelques activités fondamentales. Celles-ci incluent la spécification, la conception, l'implantation, la validation et enfin l'évolution.

2.4.2/ LA MÉTHODOLOGIE ASPECS

Comme nous l'avons expliqué à la section 2.3.2, nous basons nos travaux sur le métamodèle CRIO. La méthodologie ASPECS [Cossentino et al., 2010, Gaud, 2007] est basée sur ce métamodèle et fournit une démarche pour l'ensemble des phases liées au processus de conception de notre modèle.

Cette section présente succinctement les phases et activités composant la méthodologie ASPECS [Cossentino et al., 2010, Gaud, 2007]. Nous utiliserons cette méthodologie pour

concevoir notre modèle de simulation de foules. Les diagrammes UML issus de ce processus sont présentés dans l'annexe A. Les chapitres 4 et 5 détaillent les concepts et les choix principaux aboutissant à la proposition d'un modèle d'environnement. ASPECS est défini par trois phases que sont : (i) l'analyse des besoins ; (ii) la conception de la société d'agents ; (iii) l'implantation et le déploiement de la solution. Chaque phase a un objectif qui lui est propre et est subdivisée en activités. Ces phases et activités sont présentées par la suite dans l'ordre chronologique où elles doivent être utilisées.

2.4.2.1/ ANALYSE DU PROBLÈME

La phase d'analyse des besoins vise à décrire le problème sur la base des abstractions définies dans le domaine du problème du métamodèle CRIO (*cf.* figure 2.1). Elle est ensuite divisée en activités présentées dans la Figure 2.4 [8].

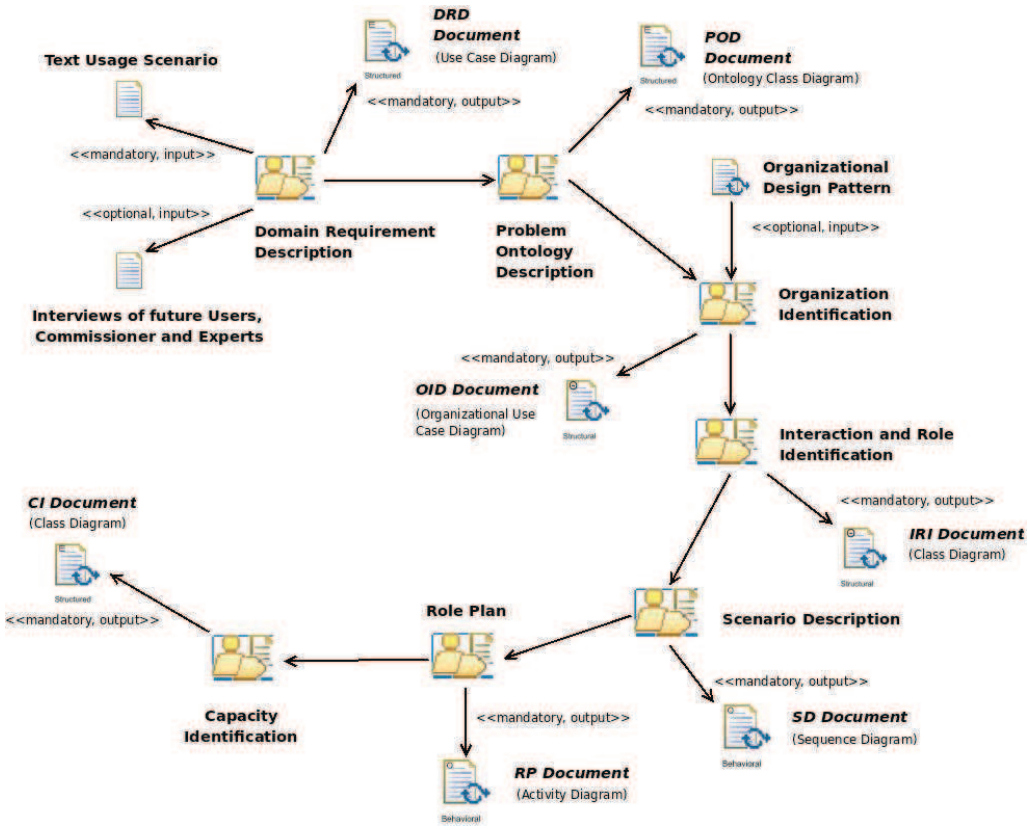


FIGURE 2.4 – Phase d'ASPECS [Cossentino et al., 2010] sur l'analyse des besoins.

Cette phase se définie par les sept activités suivantes :

1) La description des besoins du domaine identifie, classifie et hiérarchise l'ensemble des besoins fonctionnels et non-fonctionnels des différentes parties prenantes du projet. Elle donne également une première estimation du périmètre de l'application et de sa complexité.

8. http://www.aspecs.org/System_Requirements_Analysis

2) La description de l'ontologie [9] du problème met en place les concepts et actions propres au problème en donnant une définition du contexte de l'application et du vocabulaire du domaine.
3) L'identification des organisations fournit une première décomposition organisationnelle du système, où chaque organisation est liée à des objectifs.
4) L'identification des interactions et des rôles vient compléter les organisations par l'ajout des rôles nécessaires à leur fonctionnement et l'ajout des interactions entre ces dits-rôles.
5) La description des scénarios d'interactions des organisations donne lieu à des comportements de plus haut niveau. Elle décrit l'ordonnancement des rôles ainsi que la séquence d'arrivée ou de transfert des informations externes à l'organisation.
6) La description des plans de comportement des rôles spécifie le comportement de chaque rôle en fonction de ces objectifs et décrit l'ordonnancement des interactions.
7) L'identification des capacités vise à l'augmentation de la généricité des comportements des rôles via le découplage entre leur comportement et leurs dépendances externes.

2.4.2.2/ CONCEPTION D'UNE SOCIÉTÉ D'AGENTS :

La Figure 2.5 [10] montre la phase de conception de la société d'agents sur la base des abstractions définies dans le domaine agent du métamodèle CRIO (*cf.* figure 2.2). Cette phase a pour objectif la modélisation de la société d'agents dont le comportement global fournira une solution au problème décrit précédemment.

Elle se développe en une dizaine d'activités :

1) La description de l'ontologie de la solution affine l'ontologie du problème en y intégrant les connaissances relatives à la solution.
2) La description ontologique des communications décrit les communications et conversations entre les rôles.
3) La description des comportements des rôles est défini par un cycle de vie intégrant les tâches, capacités et communications qui leurs sont assignées.
4) La description des protocoles d'interaction permet de spécifier les protocoles utilisés précédemment lorsqu'ils ne correspondent à aucun des protocoles existants FIPA.
5) La description des dépendances entre organisations définit les services et ressources externes nécessaires au bon fonctionnement des rôles, ainsi que les associations avec les rôles et les capacités.
6) L'identification des agents détermine les agents composant le niveau le plus bas de la hiérarchie et leurs objectifs individuels.
7) La description des architectures des agents apporte des indications précises sur l'architecture qui devrait être adoptée par les agents.
8) L'identification des contraintes entre rôles exprime les éventuelles contraintes pouvant exister entre des rôles de différentes organisations.

9. Ici, ce terme est utilisé pour définir l'étude des propriétés générales du système à modéliser.
10. http://www.aspecs.org/Agent_Society_Design

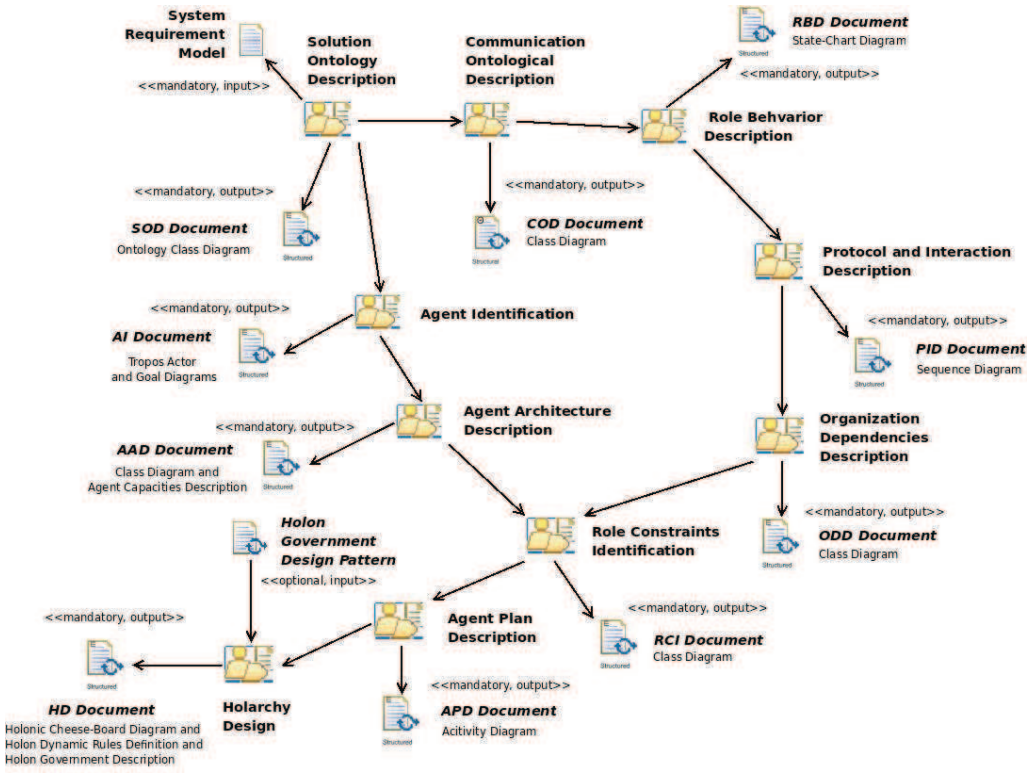


FIGURE 2.5 – Phase d'ASPECS [Cossentino et al., 2010] sur la conception d'une société d'agents.

9) La description des plans des agents détermine le plan personnel de chaque agent en fonction de ses motivations individuelles et des buts poursuivis.

10) La conception des holarchies finalise la conception de la société d'agent par une synthèse globale. Le travail des précédentes activités sont regroupées et résumées dans un travail décrivant la structure générale du système et les règles qui régiront la dynamique de cette structure.

Notons que les étapes 6 et 7 peuvent être faites en parallèles des étapes 2, 3, 4 et 5. Cette phase vise à concevoir une société d'agents dont le comportement global est en mesure de fournir une solution efficace au problème décrit dans la phase précédente tout en satisfaisant aux exigences associées (*i.e.* les hypothèses et contraintes du problème). À ce stade, le problème a été modélisé en termes d'organisations, de rôles, de capacités et d'interactions. Le résultat de cette phase de conception est un modèle de société agent impliqué dans la solution en termes d'interactions sociales et de dépendances entre entités (Holons et/ou agents). Après la première activité, le flot de conception est divisé en deux chemins alternatifs : l'un concerne les aspects sociaux et organisationnels du système (les activités comprises entre 2 et 5) alors que l'autre traite de la conception d'agents considérés comme des individus ayant des objectifs individuels (les activités 6 et 7). En d'autres termes, nous pouvons dire que les agents sont conçus comme des individus qui possèdent les capacités nécessaires pour jouer un rôle au sein d'une structure sociale (holons). Une fois l'obtention de la réalisation des capacités nécessaires pour exhiber leur comportement, les holons peuvent être considérés (à un niveau supérieur d'abstraction) comme propriétaires des capacités qu'ils ont mises en œuvre. Par consé-

quent, ils peuvent jouer un rôle dans les holons de niveau supérieur permettant ainsi la composition de l'holarchie.

2.4.2.3/ IMPLANTATION ET DÉPLOIEMENT DE LA SOLUTION :

La dernière phase (*cf.* Figure 2.6 [11]) consiste à implémenter la solution définie dans la phase précédente puis à fournir une configuration pour le déploiement du modèle. La plateforme JANUS peut être utilisée pour l'implantation et le déploiement de la solution. Cette phase est constituée de huit activités :

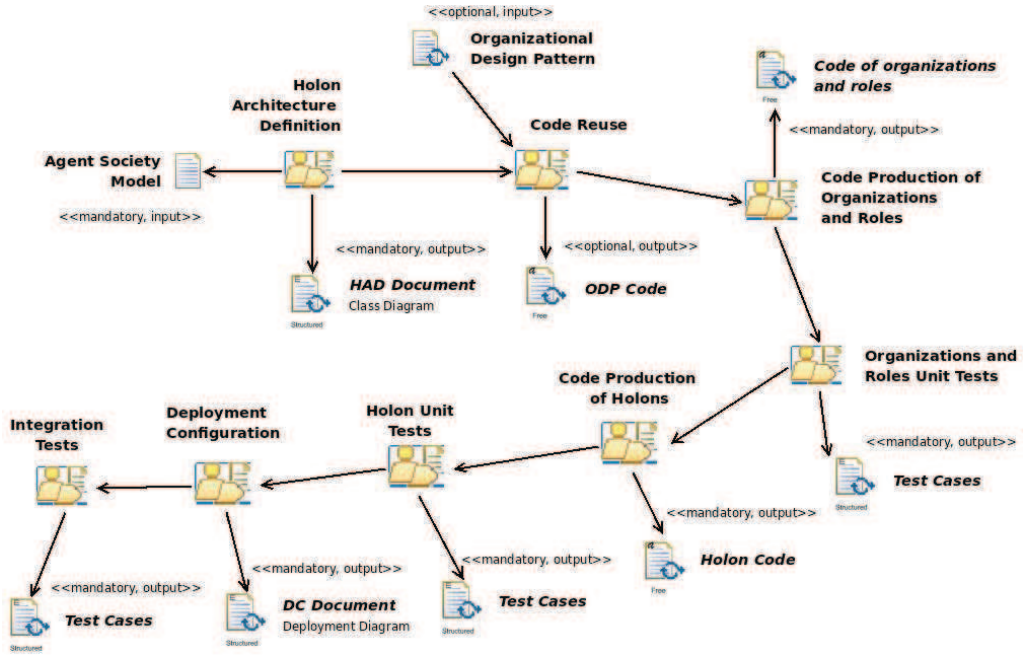


FIGURE 2.6 – Phase d'ASPECS [Cossentino et al., 2010] sur l'implantation et le déploiement.

1) La définition de l'architecture des holons porte sur ceux impliqué dans la mise en œuvre de la solution définie précédemment.
2) La ré-utilisabilité du code vise à l'intégration du code des modèles d'organisation précédemment conçus au sein de l'application en cours de développement.
3) La production du code des organisations et des rôles.
4) Les organisations et les tests unitaires des rôles permettent de tester les comportements individuels représentés par les rôles et les comportements globaux correspondant à des organisations.
5) La production du code des holons avec leur cycle de vie et les aspects de planification.
6) Les tests unitaires des holons valident le comportement de l'holon globale.
7) La configuration du déploiement détaille la manière de déployer les holons ainsi que les relations qu'ils auront avec les éléments extérieurs. Il définit la manière dont sera distribuée l'application.

11. http://www.aspecs.org/Implementation_and_Deployment

8) Les tests d'intégration vérifient si le système répond efficacement à toutes les exigences identifiées dans la solution apportée lors de la seconde phase d'ASPECS.

Les phases d'implantation et de déploiement sont essentiellement basée sur la production de code. Elles visent à implémenter et déployer la solution orientée-agents, définie dans la phase précédente, en l'adaptant à la plateforme choisie. Cette partie du travail est donc dépendante de la plateforme de déploiement.

ASPECS est une des rares approches méthodologiques permettant la conception de modèles holoniques [Isern et al., 2011] (plusieurs niveaux différents d'abstractions sont possibles). Elle permet la réutilisation et la modularité des modèles et des connaissances. Couplée à JANUS, elle permet l'implantation des organisations et des rôles comme étant des *entités de premier ordre* et où leur description est indépendante de l'architecture des agents. Le dernier point fort et non des moindres est qu'elle couvre l'intégralité des étapes de conception. Dans les chapitres 4 et 5, nous utilisons ASPECS pour concevoir un modèle pour la simulation de foule.

Nous présentons dans la section suivante les plateformes multiagents. Nous expliquons notre choix de JANUS comme plateforme d'implantation pour la simulation de foules.

2.5/ LES PLATEFORMES AGENTS

Certaines plateformes agents sont développées pour la mise en place de comportements complexes d'agents, comme SemanticAgent [de Lucena et al., 2003], Jadex [Pokahr et al., 2005] et Jason [Bordini et al., 2005, Rao, 1996]. L'explosion de la quantité d'informations échangées à travers les réseaux ont fait de ces plateformes un atout pour la modélisation de systèmes de connaissances. Un autre atout en est la création de modèles cognitifs, généralement basés sur la logique du premier ordre. D'autres plateformes sont utilisées pour le pilotage opérationnel de systèmes complexes comme DoMIS [12]. Elles sont utiles pour la gestion de projets imposants avec de nombreux acteurs ayant chacun un ensemble de compétences et de nombreuses contraintes. Enfin, des plateformes comme MASSIVE [13] et Golaem [14] sont utilisées pour la simulation de SMA dans le cadre de simulations réalistes de foules avec des effets spéciaux de scènes animées.

Afin d'assurer la mise en place d'une simulation multiniveau, nous orientons notre choix vers une plateforme ayant des concepts d'organisations. Elles sont les plus aptes à la modélisation de systèmes hiérarchiques pour la simulation multiniveau. Nous ajoutons à notre comparatif, la plateforme JADE qui est un standard. Notre comparatif se base sur celui effectué par [Noel, 2012] et porte sur les plateformes suivantes : JADE, JANUS, Madkit, MAGIQUE et J-MOISE+.

– JADE (Java Agent DEvelopment) est un framework de développement de SMA. Il offre en particulier un support avancé de la norme FIPA-ACL [FIPA, 1998], ainsi que des outils de validation syntaxique des messages entre les agents basés sur les ontologies.
– JANUS [Gaud et al., 2008a] est une plateforme multiagent modulaire. Elle propose une implantation du métamodèle CRIO pour l'implantation de systèmes multiagents organisationnels et holoniques. Le métamodèle de JANUS est un raffinement du métamodèle

12. http://iegd.institut.online.fr/B-ADSc-intelligence-systeme-fr.htm
13. http://massivesoftware.com/index.html
14. http://www.golaem.com

CRIO. Ainsi, la plateforme minimise l'écart entre le modèle et son implantation. La plateforme JANUS est une plateforme généraliste fournissant également des extensions logicielles dédiées à la simulation. Deux extensions proposent des modèles d'environnement physique (au sens de [Odell et al., 2002]) et de leur exécution au sein d'une simulation. La première, nommée JAAK, propose un modèle d'environnement discret basé sur une grille en deux dimensions. La seconde extension, nommée JASIM, propose un modèle d'environnement continu et entièrement en trois dimensions. Ces deux extensions utilisent le modèle d'interaction Influence-Réaction [Ferber et Müller, 1996].
- MadKit est une plateforme multiagent modulaire et généraliste. Elle propose une implantation du métamodèle Agent/Groupe/Rôle [Ferber et al., 2004]. Cette plateforme fournit, entre autre, l'extension TurtleKit qui propose un modèle d'environnement discret basé sur une grille en deux dimensions.
- MAGIQUE [Mathieu et Routier, 2001] une plateforme pour les agents physiquement distribués, fournissant un modèle de communication d'appel à *la cantonade*[15]. Dans MAGIQUE, les compétences sont dissociées des agents. Elles sont ensuite greffées comme plugin dans les agents au gré du concepteur.
- J-MOISE+ est une plateforme couplant les modèles MOISE+ et Jason. MOISE+ [Hannoun et al., 2000] est un modèle organisationnel pour les SMA. Jason est un environnement de développement d'agents dans le formalisme AgentSpeak [Rao, 1996].

Étant donné l'approche utilisée pour nos travaux et les concepts qui nous sont nécessaires, nous comparons les dites-plateformes en utilisant les critères et les notations décrits ci-dessous.

Organisationnel : '+' si les agents possèdent des rôles ; '++' si les rôles sont situés au sein d'organisations ; '+++' si les organisations sont hiérarchiques

Hiérarchique : '+' si les comportements d'agents sont décomposables ; '++' si la notion de capacité (ou compétence) est ajoutée.

Dynamique des rôles : '+' si les agents peuvent changer de rôles ; '++' si le rôle est implémenter comme une « entité de première classe »

Communications inter-agents : '+' si un langage ACL [FIPA, 1998] (de l'anglais : *Agent Communication Language*) est utilisé ; '++' si, en plus, un protocole de communication inter-agents est défini.

Environnement situé : '+' si il existe un environnement physique situé au moins 2D et discret ; '++' si l'environnement est 3D et continu.

Politique d'exécution : '+' si le modèle créé est multithread, '++' si la politique d'exécution est définissable, '+++' si cette politique est adaptable (exemple : les réponses sont données en temps voulu).

Ces choix sont basés sur les points importants quant à la modélisation d'un système hiérarchique pour la simulation multiniveau. En effet, nous souhaitons utiliser un modèle organisationnel hiérarchique possédant une certaine liberté quant à la mise en place de comportements hiérarchiques et complexes, mais également en terme de politique

15. Ici, l'appel à la cantonade signifie qu'un agent communique avec tous les autres agents sans s'adresser à un agent en particulier.

d'exécution. Étant dans la modélisation d'un simulateur de foule, la modélisation d'un environnement virtuel reste nécessaire. Ainsi, l'environnement situé constitue un critère, certes secondaire, dans le choix de notre plateforme.

Plateformes	Organisationnel	Hiérarchique	Dynamique des rôles	Communications	Environnement situé	Politique d'exécution
JADE	*n/a*	*n/a*	++	++	*n/a*	++
JANUS	+++	++	++	+	++	+++
MadKit	++	++	+	+	+	+
MAGIQUE	+	++	++	+	*n/a*	+
J-MOISE+	++	*n/a*	++	++	*n/a*	++

TABLE 2.1 – Récapitulatif de la comparaison des plateformes JADE, JANUS, Madkit, MAGIQUE et J-MOISE+.

Comme le montre la table 2.1, JANUS représente une plateforme adaptée à la réalisation de nos travaux, car modulaire et laissant libre cours à la modélisation d'holarchies et leurs politiques d'exécutions complexes au sein d'holons. JADE et Madkit sont des plateformes souvent utilisées dans les projets industriels et académiques. Toutefois, elles permettent difficilement la mise en place de systèmes holoniques. Concernant MAGIQUE, bien que sa prise en main soit simple et qu'elle intègre le concept de compétence, elle ne supporte pas nativement le concept d'environnement physique. De plus, cette plateforme ne permet pas de mettre en place des politiques d'exécutions complexes, essentielles pour la simulation multiniveau.

2.6/ CONCLUSION

Ce chapitre a permis de présenter les Systèmes MultiAgents (SMA) et leur extension aux holarchies. Les holarchies sont basées sur un ensemble d'holons, agents particuliers pouvant être composés (membres) d'autres holons, interagissant entre eux pour la résolution de tâches complexes.

Cette propriété hiérarchique, ainsi que la dynamique intrinsèque d'une holarchie, favorise la modélisation et la simulation de systèmes hiérarchiques, comme les simulations de foules dans des environnements situés. En effet, la foule peut être représentée par des flux, contenant des groupes qui contiennent à leur tour des piétons. Simuler des foules revient à simuler les piétons la composant et les groupes qu'ils forment. Nous retrouvons donc des propriétés hiérarchiques nécessaires à la représentation des piétons et des groupes. De plus, les groupes changent durant la simulation. Ainsi, l'utilisation d'une holarchie permet de fournir : les propriétés hiérarchiques, la dynamique des groupes, et les différents points de vue du système (ici, les piétons et les groupes).

CRIO permet de modéliser ces systèmes complexes par le biais des concepts tels que l'organisation, le rôle, l'interaction et la capacité. Les agents jouent des rôles qui sont définis au sein d'organisations. Le comportement des agents est ainsi défini par ces rôles et leurs interactions. La capacité est la représentation d'un savoir faire personnel d'un agent. Des capacités peuvent être requises pour jouer un rôle. De plus, une organisation peut exhiber certaines capacités à partir de ses rôles. Cette dualité de la capacité permet de faire le lien entre deux niveaux adjacents de la hiérarchie organisationnelle en liant un

rôle à une organisation. Une partie du rôle est alors raffinée par une organisation.

Dans ce chapitre, nous avons également détaillé la méthodologie et la plateforme que nous utiliserons conjointement durant le processus de conception de notre modèle de simulation de foules dans un environnement intérieur. Nous avons procédé à une présentation succincte des principes d'une méthodologie orientée-agents et détaillé ASPECS qui nous servira pour la conception de notre modèle dans les chapitres 4 et 5.

Enfin, nous avons comparé quelques plateformes multiagents. Cette comparaison se base sur les critères permettant la modélisation et la simulation de systèmes holoniques : support de la hiérarchie du système, approche organisationnelle, définition et configuration des politiques d'exécution des agents, *etc*. La platerforme sélectionnée lors de la conception de notre modèle est JANUS, qui implante le métamodèle organisationnel CRIO et propose des outils pour la simulation des SMAH.

3

MODÈLES DE SIMULATION DE FOULES

LES travaux présentés dans cette thèse portent sur la conception d'un modèle de simulation de foules dans des environnements intérieurs de bâtiments.
Ces travaux sont situés à l'intersection de plusieurs domaines scientifiques complémentaires : la simulation de systèmes complexes, la simulation multiniveau, la Simulation Orientée-Agents (SOA) et la simulation de foules.
Ce chapitre présente une synthèse de la littérature associée à ces différents domaines. Il se focalise particulièrement sur les travaux portant sur la simulation de foules dans des environnements intérieurs de bâtiments.

Sommaire

3.1/ DÉFINITION DE LA SIMULATION

[Shannon, 1977] définit la simulation comme *« le processus permettant de concevoir un modèle d'un système réel et de mener des expérimentations sur la base de ce modèle pour comprendre le comportement du système ou évaluer différentes stratégies pour son fonctionnement (dans les limites imposées par un critère ou un ensemble de critères) »* .

L'objectif de la simulation est de faciliter la compréhension de la dynamique d'un système et tenter d'en prédire l'évolution. Satisfaire cet objectif nécessite l'élaboration d'un modèle du système à étudier, son exécution sur un calculateur, et l'analyse des résultats de cette exécution [Fishwick, 1997].

La simulation du système applique l'ensemble des mécanismes qui gèrent les changements d'état d'un modèle du système. Le modèle du système correspond à l'ensemble des lois, conditions ou contraintes qui définissent le comportement du système, ainsi que la manière dont ses composantes sont agrégées.

L'exécution, quant à elle, doit faire évoluer dans le temps le modèle du système [Coquillard et Hill, 1997]. Pour y parvenir, elle est généralement associée à un ensemble d'outils qui constituent le simulateur.

Dès lors que l'on considère un système et son modèle, la question de la cohérence de ce modèle se pose. Dans la même optique, dès lors que l'on considère un modèle et le simulateur en charge de l'exécuter, la cohérence de ce simulateur se doit d'être étudiée. Dans [Zeigler et al., 2000], deux relations fondamentales sont distinguées. Elles doivent être prises en compte lors de la conception d'une simulation :

Relation Système-Modèle : La relation de modélisation définit la notion de validité de l'expérience en soulevant la question de la correspondance entre le modèle (et le comportement qu'il génère) et le système source dans le contexte du cadre expérimental. Il s'agit donc de déterminer si le modèle du système est une simplification acceptable de celui-ci en fonction des critères qualitatifs choisis et des objectifs de l'expérimentation. Cette relation se rapporte directement à la cohérence du modèle de simulation.

Relation Modèle-Simulateur : Cette relation s'intéresse à l'aptitude du simulateur à reproduire fidèlement la dynamique du modèle initial, et à gérer correctement les phases de changement d'états du modèle.

La précision des résultats de la simulation est un indicateur permettant de quantifier la relation Système-Modèle. Elle est définie dans la norme [ISO-3534-1, 2006] par *« l'étroitesse de l'accord entre le résultat d'essai et la valeur de référence acceptée. Elle est déterminée par la justesse et la fidélité. »* Sa définition concrète, sous forme d'équations par exemple, est dépendante du domaine d'application de la simulation et du modèle utilisé. Dans la suite de cette thèse, nous considérons qu'il y a une corrélation entre l'évolution de la précision et le "réalisme" du modèle de simulation : plus le modèle est précis, plus il est proche du système source. Dans le chapitre 5, nous proposons un cadre permettant d'évaluer cette précision pour des simulations multiniveaux de foules dans des environnements intérieurs.

Plusieurs niveaux d'observation peuvent être utilisés pour modéliser un système [Davidsson, 2001, Hoogendoorn et Bovy, 2001]. Ces niveaux d'observation sont souvent appelés les échelles du modèle. Différents niveaux d'observation seront détaillés dans la suite de ce chapitre.

La relation entre la précision et l'échelle d'un modèle de simulation est importante : plus l'échelle est petite, plus la précision est grande. Le corollaire communément admis de cette relation est que plus l'échelle est petite, plus les ressources de calcul nécessaires pour réaliser la simulation sont importantes.

3.2/ SIMULATION DE SYSTÈMES COMPLEXES

Dans le cadre de la simulation des systèmes complexes, et plus spécifiquement celle de la dynamique de populations composées d'individus en interaction, deux grands types de simulations peuvent être distingués en fonction du niveau d'abstraction de leurs modèles [Davidsson, 2001, Hoogendoorn et Bovy, 2001] :

Les simulations macroscopiques (ou macro-simulation) : Elles sont généralement basées sur des modèles mathématiques (équations, *etc.*), lesquels sont établis à partir des caractéristiques moyennées d'une population d'individus. Dans ce type de simulation, la population est considérée comme une structure qui peut être caractérisée par un certain nombre de variables. Le modèle de la simulation tente d'analyser les variations et les changements de ces variables. La perspective adoptée par la simulation macroscopique se situe au niveau du système dont elle cherche à recréer la dynamique globale.

Les simulations microscopiques (ou micro-simulation) : Elles tentent explicitement de modéliser les comportements spécifiques des individus composant une population. La perspective adoptée se situe au niveau de l'individu ; la dynamique du système est issue des interactions entre les individus. La structure du système est considérée comme émergente de ces interactions. Les modèles de simulations microscopiques sont généralement basés sur les automates cellulaires ou les Systèmes MultiAgents (SMA).

Les forces et les faiblesses de ces deux approches ont été soulignées par [Van Dyke Parunak et al., 1998] qui considèrent que les approches microscopiques et plus particulièrement multiagents sont plus appropriées aux systèmes caractérisés par un haut degré de localisation et de distribution. [Van Dyke Parunak et al., 1998] considèrent également que les approches microscopiques sont régies par un processus de décision discret, alors que les approches macroscopiques sont naturellement appliquées à des systèmes qui peuvent être modélisés de manière centralisée et dans lesquels la dynamique est régie par des lois physiques.

De manière plus générale, le volume et la précision des données requises pour l'initialisation des approches macroscopiques sont beaucoup plus réduits que pour les approches microscopiques. De plus les approches macroscopiques nécessitent moins de ressources de calcul. Dès lors, les approches macroscopiques apparaissent comme les plus aptes à simuler des systèmes de grande échelle. En revanche, elles s'avèrent inadéquates, voire incapables de modéliser l'impact d'événements microscopiques.

Les approches microscopiques quant à elles peuvent aisément modéliser l'impact de tels événements. Elles permettent de comprendre le fonctionnement d'un système puisque la structure du système émerge des interactions entre les individus qui le composent. En revanche, elles nécessitent des données précises et nombreuses pour leur initialisation, ce qui les rend de fait difficilement applicables à des systèmes de grande échelle.

Ce constat, partagé également par [Magne et al., 2000] pour la simulation des réseaux de transport, amène naturellement vers des modèles qui tentent d'allier les avantages de ces deux approches. Deux perspectives peuvent alors être envisagées :
– Tout d'abord, il est possible de disposer de simulations dont le niveau de détails se situe à une échelle intermédiaire entre ces deux extrémités, tel que le niveau mésoscopique où l'individu peut être modélisé, mais pas ses interactions.
– Il est également possible de combiner différents niveaux au sein de la même simulation. Les approches hybrides réunissent en général deux ou trois niveaux.

La simulation multiniveau, quant à elle, peut être considérée comme une généralisation de ces deux perspectives où plusieurs niveaux d'abstraction cohabitent au sein de la même simulation et où les transitions entre ces niveaux peuvent être assurées de manière dynamique.

3.3/ SIMULATION MULTINIVEAU

La simulation multiniveau permet de faire cohabiter différents niveaux d'abstraction au sein d'une même exécution et d'assurer la transition dynamique entre eux en fonction de contraintes définies (dépendantes du modèle ou du cadre expérimental) [Gaud, 2007].

De nombreux champs d'application s'intéressent à la simulation multiniveau, comme la sociologie [Sawyer, 2003, Schillo et al., 2001], l'éthologie [Morvan et al., 2007], la biologie cellulaire [Lepagnot et Hutzler, 2009, Zhang et al., 2007, 2009], la simulation de foules [Anh et al., 2012, Chenney et al., 2001].

Les sections qui suivent présentent les définitions et concepts liés à la simulation multiniveau ainsi qu'un état de l'art succinct des modèles multiniveaux, respectivement.

3.3.1/ DÉFINITIONS ET CONCEPTS

[Ducrot et al., 2010] regroupent les concepts clés "*niveau*" et "*échelle*" en trois notions complémentaires que sont les notions de *dimension*, de *niveau* et de *résolution*.

Un système peut se décrire par de multiples dimensions. Celles-ci peuvent être considérées sur plusieurs niveaux. Ces niveaux constituent les hiérarchies pour différentier les représentations d'un système et sa décomposition en sous-systèmes sur une dimension prédéfinie. La résolution correspond au degré de précision de la description ou de la perception du système. Elle permet d'affiner la notion de niveau. Les notions de dimension, de niveau et de résolution s'imbriquent telle une hiérarchie décrivant un système. Une dimensions peut considérée plusieurs niveaux, qui à leur tour peuvent considérés plusieurs résolutions.

Dans nos travaux, nous considérerons uniquement une dimension spatiale et temporelle. Toutefois, la dimension temporelle sera considérée sur un unique niveau et une unique résolution. Nous abordons donc la simulation multiniveau du point de vue spatial. Quant à la résolution, elle correspond au degré de précision de la description ou de la perception du système (*e.g.* l'approximation des distances, la perception d'individus et de groupes, *etc.*).

Nous considérons une simulation multiniveau d'un système, comme étant l'application

de la dynamique du système sur un niveau de représentation de sa hiérarchie et ce sur une dimension spatiale. Les dynamiques sont propres au niveau de représentation. La simulation multiniveau de foules reviendrait donc à choisir un niveau de représentation des entités composant le système, auquel sera appliqué les dynamiques du système pour une résolution donnée. Dans la suite du mémoire, nous considérons que chaque niveau possède une unique résolution.

3.3.2/ MODÈLES MULTINIVEAUX

Cette section propose un bref tour d'horizon des approches existantes dans le domaine de la simulation multiniveau. Deux niveaux sont très souvent considérés : microscopique et macroscopique [Anh et al., 2012] ou microscopique et mésoscopique [Burghout et al., 2005]. Ces modèles sont généralement dépendants de l'application cible.

Dans de nombreuses approches, le système est divisé en sous-systèmes où le niveau de simulation de chaque sous-système est fixé a priori durant la simulation. Les transitions entre ces sous-systèmes sont donc effectuées à des points de connexion déterminés [Anh et al., 2012].

Dans d'autres approches, le niveau de simulation de l'ensemble du système varie durant l'exécution. Ce point de vue est d'ailleurs partagé par [Ghosh, 1986] qui propose l'un des premiers modèles de simulation multiniveau dynamique. Son domaine d'application est la simulation de composants électroniques. Il utilise un modèle basé sur la décomposition hiérarchique des composants, dans lequel le niveau de décomposition peut changer dynamiquement. Ce niveau de décomposition correspond à un niveau de simulation. Mais le niveau n'est pas automatiquement déterminé en fonction des contraintes de la simulation ou des conditions d'applicabilité du niveau de simulation. C'est l'utilisateur qui choisit le niveau de décomposition durant la simulation.

Le domaine de la simulation de foules fournit des modèles avec davantage de dynamiques [da Silveira et Musse, 2006a, Donikian, 1997a, Thomas, 1999]. Dans [Musse et Thalmann, 2001], la notion de niveau d'autonomie pour la simulation de personnages et de foules est proposée. Ils distinguent trois niveaux d'autonomie où le comportement de l'entité est soit fixe, soit *« autonome »* (proche des agents réactifs simples dans les SMA), soit directement contrôlé par l'utilisateur. Une autre contribution de ces travaux concerne la modélisation de la structure d'une foule d'individus qui est décomposée hiérarchiquement sous forme de groupes. L'objectif de ces travaux consiste à assurer le meilleur niveau de réalisme visuel pour la simulation, tout en maintenant des performances temps-réel [1]. En revanche, le niveau de précision des comportements des entités est relativement faible face à celui atteignable avec des outils comme les SMA. Cependant, le principe de l'approche constitue l'une de nos inspirations.

Dans la même optique, [Brogan et Hodgins, 2002] ont adapté la notion de niveau de détails (*"level of detail"*), initialement utilisée pour moduler la complexité de la représentation géométrique d'un environnement intérieur, aux comportements des entités évoluant dans cet environnement. Toujours dans le domaine de la simulation de foules, les travaux de [Chenney et al., 2001, Chenney et Forsyth, 1997] visent également à maintenir un niveau de réalisme maximal dans une simulation tout en conservant des performances

1. Ici *« temps réel »* se réfère à la simulation d'événements dont la vitesse de déroulement dans l'espace simulé est équivalente à celle perçue dans le monde réel.

optimales. Ils introduisent notamment la notion de *"proxy simulation"* où les entités hors du champ de vision de l'utilisateur sont simulées à un faible niveau de détails, sous forme événementielle. Les transitions dynamiques sont assurées de sorte à re-générer dans un état cohérent des entités qui vont apparaître dans le champ de vision de l'utilisateur.

Dans le domaine de la simulation de réseaux de transports, les approches hybrides mettant en évidence deux niveaux d'abstraction *« micro-macro »* [Magne et al., 2000] et *« méso-micro »* [Burghout, 2004, Burghout et al., 2004, 2005, Ziliaskopoulos et al., 2006] ont été proposées. Ces approches se basent généralement sur un graphe pour représenter le réseau routier, dont les routes sont des arcs et les intersections sont des nœuds. Les entités situées sur les routes sont simulées suivant une approche avec un niveau moins précis que celles situées sur les nœuds.

Dans la simulation de déplacement de piétons, d'autres auteurs proposent d'utiliser des modèles pour calculer les mouvements de piétons et de groupes de piétons. [Gloor et al., 2004] proposent un modèle hybride pour la simulation de piétons dans des environnements de grande taille. [Gaud et al., 2008b] utilisent le même modèle de déplacement à tous les niveaux d'abstraction. Ils introduisent des indicateurs énergétiques pour évaluer la qualité de l'approximation de la simulation des comportements des piétons à des niveaux d'abstraction mésoscopiques. [Razavi et al., 2011a] regroupent les corps des piétons sous la forme d'une hiérarchie, pour ensuite calculer le déplacement des entités par la méthode de *« Fast Multipole Method »* . Ce modèle permet de réduire la complexité algorithmique des calculs des déplacements des entités et de fournir une estimation précise de l'erreur d'approximation.

[Camus et al., 2012] proposent un modèle macro-micro pour le déplacement d'oiseaux, fondé sur le modèle de déplacement de nuées proposé dans [Wilensky, 1998]. Ces deux modèles peuvent être adaptés à la simulation de déplacements de piétons [Camus et al., 2012].

Dans les modèles de simulations de foules, certains modèles se distinguent par l'utilisation d'environnements virtuels complexes [da Silveira et Musse, 2006b, Farenc, 2001, Tecchia et al., 2001, Thomas, 1999]. Ces environnements sont soit multidimensionnels, soit hiérarchiques. Les modèles multidimensionnels comme ceux de [da Silveira et Musse, 2006b, Tecchia et al., 2001] permettent d'avoir plusieurs couches d'informations différentes, alors que les modèles hiérarchiques [Farenc, 2001, Thomas, 1999] se basent sur la structuration de l'environnement en différents niveaux. Ces types de modèles sont un premier pas vers la simulation multiniveau, car ils intègrent les notions de dimension ou de niveau (pour les modèles hiérarchiques). C'est sur l'extension de modèles d'environnement hiérarchique pour une simulation multiniveau que portent nos travaux.

3.4/ SIMULATION ORIENTÉE-AGENTS

La Simulation Orientée-Agents (SOA), ou simulation multiagent, se réfère aux modèles individu-centrés [Amblard, 2003] et fournit un outil permettant de modéliser et simuler la dynamique de populations composées d'individus en interaction. Ce type de simulation assimile généralement l'individu à un agent. La SOA a été appliquée dans plusieurs domaines tels que la robotique [Drogoul, 1993, Kitano et al., 1997], l'éthologie [Drogoul et Picault, 1999], l'écologie et la biologie (*cf.* CORMAS, GAEMAS et Mobidyc), ou les sciences

sociales [Conte et al., 1998, Gilbert et Troitzsch, 2005].

Le modèle proposé par [Michel, 2004] pour la modélisation et la simulation de SMA constitue une des principales inspirations des travaux décrits dans ce manuscrit. Il adopte une approche multivue et distingue quatre aspects fondamentaux dans un modèle de SOA :

Comportements : cet aspect traite de la modélisation des processus de délibération des agents (leurs « *esprits* »).

Environnement : ce point de vue vise à définir les différents objets physiques du monde simulé (l'environnement situé et le « *corps* » des agents) ainsi que la dynamique endogène de l'environnement.

Ordonnancement : cet aspect concerne la modélisation de l'écoulement du temps et la définition de l'ordonnancement utilisé pour exécuter le comportement des agents.

Interaction : cette vue s'intéresse plus particulièrement à la modélisation du résultat des actions et des interactions à un instant donné.

La figure 3.1 illustre les relations entre ces quatre aspects fondamentaux d'une SOA.

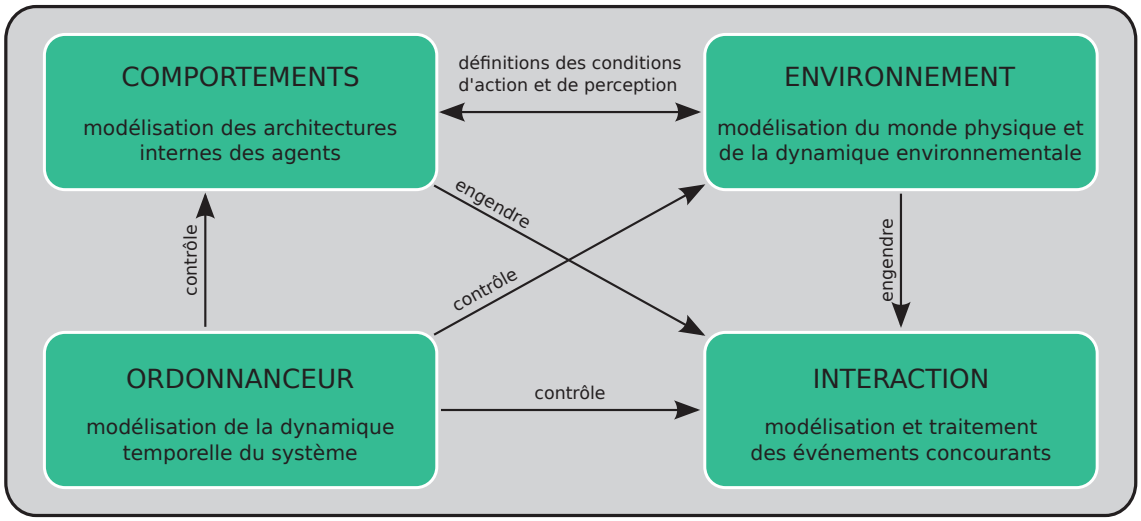


FIGURE 3.1 – Les quatre aspects d'un modèle de SOA selon [Michel, 2004]

La modélisation et l'implantation de chacun de ces aspects et de leurs relations sont autant de points délicats qui soulèvent les problématiques suivantes :

Respecter la contrainte de localité : un agent est une entité dont les perceptions et les actions n'ont qu'une portée locale. Deux approches principales existent pour respecter cette contrainte :

- L'approche discrète (centrée environnement) où la discrétisation de l'environnement sous forme de zones définit la granularité des perceptions et des actions des agents.
- L'approche continue (centrée agents) où la portée de chaque perception et de chaque action fait l'objet d'un traitement particulier qui est fonction de la nature et des caractéristiques de l'agent concerné [Weyns et al., 2004].

Ces deux approches peuvent également être combinées.

Respecter la contrainte d'intégrité environnementale : un agent ne doit pas être en mesure de modifier directement les variables d'état de l'environnement.

Biais de simulation et simultanéité des actions : Pour éviter d'introduire des biais dans la simulation, il est nécessaire de disposer d'un modèle de gestion de l'action des agents et du temps, qui permettent de modéliser la simultanéité de deux événements. Un modèle de simulation ne doit pas être lié à une implantation particulière [Zeigler et al., 2000]. L'ordre dans lequel les agents sont activés influe sur la dynamique du système et peut entraîner des biais de simulation.

De manière générale, pour prendre en considération ces différentes contraintes et concevoir des SOA, nous adoptons le modèle Influence-Réaction [Ferber et Müller, 1996, Michel, 2001, 2006, Weyns et Holvoet, 2004a].

L'action d'un agent au sein d'une simulation est généralement modélisée comme la transformation d'un état global [Ferber, 1995], c'est-à-dire la modification directe d'une des variables de l'environnement. Le cœur du modèle Influence-Réaction consiste à séparer l'action d'un agent de l'effet qu'elle produit. En effet, un agent produit des influences sur son environnement et non des actions au sens vu précédemment. L'influence représente le désir d'un agent de voir modifier l'environnement d'une certaine façon. Le résultat effectif de cette tentative de modification de l'environnement par un agent ne peut être calculé sans connaître l'ensemble des influences produites au même instant. Ce modèle se base en fait sur la distinction claire entre deux dynamiques qui sont combinées dans un SMA [Michel, 2006] : (i) La dynamique au niveau de l'agent qui produit des influences. (ii) La dynamique au niveau du système qui calcule la réaction de l'environnement compte tenu de l'ensemble des influences émises simultanément. Pour calculer cette réaction, les influences sont considérées en fonction des lois de l'univers [Ferber et Müller, 1996].

Adopter le modèle Influence-Réaction implique également de distinguer deux composantes de l'agent : son système décisionnel et ses composantes physiques [Michel, 2004, p. 117-119]. Cette distinction est nécessaire afin de clairement séparer le modèle d'environnement du modèle du système. En effet, le système décisionnel de l'agent est dépendant du système cible alors que ses composantes physiques sont intégrées au modèle d'environnement. Cette distinction correspond à la partition entre les variables utilisées par le système décisionnel de l'agent sur lesquelles l'agent dispose d'un contrôle total, et les variables liées à la modélisation de sa partie physique contrôlées par l'environnement et sur lesquelles l'agent n'a aucun pouvoir décisionnel.

Les SOA conduisent souvent à l'émergence de groupes locaux d'entités [Servat et al., 1998], mais fournissent rarement les moyens de les manipuler. Exploiter pleinement ces simulations implique certes la création dynamique de tels groupes d'agents, mais également leur "*agentification*" de sorte à pouvoir gérer des comportements spécifiques à chaque niveau d'abstraction. Dès lors, les systèmes multiagents hiérarchiques ou holoniques apparaissent comme une solution intéressante. [Van Aeken, 1999] introduit la notion de *systèmes multiagents minimaux* pour étudier la dynamique organisationnelle des SMA. Cette approche est basée sur la création d'un agent composé pour chaque couple d'agents atomiques qui pourraient être fusionnés. Il construit ainsi une arborescence binaire par regroupements successifs d'agents. Il fournit ensuite un ensemble de mesures pour caractériser la structure et les propriétés d'un *système multiagent minimal* telles que la taille, l'équilibre, l'entropie et la notion de distance entre systèmes. L'inconvénient principal de cette approche est d'abord lié à sa structure uniquement binaire qui conduit à la création d'arbre de grande taille et, par conséquent, à de nombreux agents intermédiaires lorsque le système est composé de nombreux individus. Ce phénomène est généralement présent pour les systèmes complexes et induit de fait un surcoût notoire

dans l'exécution des modèles. Toutefois, si cette approche demeure relativement abstraite et difficile à appliquer dans le cadre d'applications réelles, elle peut être considérée comme similaire à l'approche holonique adoptée dans ce manuscrit. L'un des avantages de la structure holonique est qu'elle n'impose pas cette partition binaire et permet de créer des groupes d'agents de tailles variables en fonction du problème et de sa dynamique. De manière générale, une hiérarchie holonique (ou holarchie) est une structure beaucoup plus souple qu'une hiérarchie.

3.5/ MODÈLES DE SIMULATION DE FOULES

Dans cette section, nous présentons les modèles de simulation de foules. Ces modèles tentent de reproduire les comportements de déplacement d'individus. Ils sont utilisés dans de nombreux domaines, telles que les sciences du transport [LU, 2007, Shiwakoti et al., 2011], les sciences de la sécurité [Braun et al., 2005b], la sociologie [Jager et al., 2001], la physique [Farkas et al., 2002, Helbing et Molnar, 1995], l'animation graphique 3D [Jiang et al., 2010, Musse et Thalmann, 1997, Reynolds, 1987], les systèmes de formation [Bottaci, 1995] ou encore l'architecture [Schelhorn et al., 1999, Turner et Penn, 2002].

Les deux domaines d'applications les plus importants, dans lesquels la simulation de foule est utilisée, sont les domaines des sciences de la sécurité et de l'architecture [Thalmann et Musse, 2007]. L'application dominante est la simulation d'évacuations dans laquelle les déplacements d'un grand nombre d'individus sont simulés dans un environnement intérieur comme les bâtiments [Braun et al., 2005a, Thompson et E., 1995], les métros [Hareesh, 2000], les bateaux [Klüpfel et al., 2000], les avions [Owen et al., 1998], les stades [Still, 2000] ou les terminaux d'aéroport [Szymanezyk et al., 2011]. L'objectif de ces simulations est d'aider les décideurs et les experts à comprendre la *« relation entre l'organisation de l'espace et les comportements humains »* [Okazaki et Matshushita, 1993].

Les modèles de simulation de foules sont généralement classés en deux catégories : les modèles macroscopiques et microscopiques [Lemercier, 2012]. Ces deux échelles sont respectivement présentées dans les sections qui suivent.

3.5.1/ SIMULATIONS MACROSCOPIQUES

Les modèles de simulations macroscopiques sont basés sur les relations déterministes de l'écoulement, la vitesse et la densité de la population (les flux de personnes ou le trafic) [Helbing et Treiber, 1998]. Ils ont étés initialement développés pour l'étude des flux, tels que des systèmes de transport, les flux intra-bâtiments, les réseaux d'une grille de surface, *etc*.

Un modèle macroscopique modélise la structure de l'espace pouvant contenir les individus et un ou plusieurs flux s'écoulant dans cette structure. La dynamique de ces flux est pour la plupart du temps représentée par un ensemble d'équations dynamiques [Helbing, 1998, Hughes, 2002]. Les contraintes associées à ces équations correspondent soient à des aspects de l'environnement (*e.g.* sa topologie), soient à des aspects de la foule (*e.g.* l'objectif de la foule). Ces contraintes et ces critères permettent de donner une certaine "intelligence" à la foule.

La représentation du système utilise une échelle généralement très élevée, impliquant une simulation peu précise. De plus, ces modèles ont généralement une représentation pauvre de l'environnement. En effet, l'environnement est souvent réduit a un contenant (dont seule la forme géométrique prime) dans lequel s'écoule un ou plusieurs fluides, modélisant la foule.

De manière générale, la simulation macroscopique ne prend pas en compte de manière fine les interactions entre les entités du système.

3.5.2/ Simulations Microscopiques

Les modèles microscopiques se focalisent sur la simulation des individus et de leurs interactions au sein d'un système. De part leur nature à modéliser les interactions endogènes au système, ils fournissent des résultats de simulation plus précis que ceux donnés par des simulations macroscopiques. Toutefois, leur coût en terme d'utilisation de ressources de calcul est plus élevé. Ces modèles ont la particularité d'être décomposés en deux sous-systèmes. L'un concerne le comportement des piétons et l'autre concerne l'environnement dans lequel les piétons évoluent.

Le comportement d'un piéton est un processus de prise de décision complexe basé sur différents niveaux d'analyses. [Daamen, 2004] considère trois niveaux distincts :
- **Le niveau stratégique** porte sur les décisions prises par le piéton pour organiser ses activités.
- **Le niveau tactique** relève de la planification de route suivant la topologie des lieux.
- **Le niveau opérationnel** s'intéresse au déplacement du piéton à court terme et aux interactions avec les autres piétons à proximité [Timmermans et al., 1992].

Nos travaux se focalisent principalement sur l'environnement associé à des comportement au niveau opérationnel. Pour fournir un modèle de simulation de foules, nous utilisons principalement des comportements de piéton basés sur le niveau opérationnel. Le lecteur intéressé par les deux autres niveaux peut se référer à [Lemercier, 2012]. Le comportement des piétons impacte sur la modélisation de l'environnement. Cette dernière revêt donc un enjeux crucial dans les simulations de foules.

La suite de cette section se focalise sur les modèles de simulations de foules dont le comportement des entités est défini au niveau opérationnel. Ces modèles de comportement peuvent se classer en trois types : les automates cellulaires, les modèles à base de forces et les modèles basés sur la vision [Moussaïd, 2010]. Nous présentons également plus en détail les modèles d'environnements associés à ces comportements.

3.5.2.1/ Automates Cellulaires

Un automate cellulaire est composé d'une grille régulière de cellules contenant chacune un "*état*" choisi parmi un ensemble fini. L'état d'une cellule varie au cours du temps en fonction des états des cellules voisines. Les variations des états dépendent d'un ensemble de règles appliquées simultanément sur toutes les cellules.

Dans le cadre des déplacements de piétons, chaque piéton occupe une unique cellule et une cellule ne peut pas être occupée par plus d'un piéton. Le comportement des piétons est normalisé par une série de règles comportementales décrivant les modalités

de passage d'une cellule à une autre.

Parmi les modèles intégrant les règles les plus simples, nous pouvons citer celui de [Fukui et Ishibashi, 1999a,b]. Chaque piéton se déplace dans la direction qu'il souhaite aller, si et seulement si la cellule (celle dans laquelle il souhaite se rendre) est libre. Si cette cellule est occupée, il tente un déplacement latéral aléatoire vers une cellule libre. Si aucun mouvement n'est possible, le piéton reste immobile.

Certains modèles se basent sur celui de [Fukui et Ishibashi, 1999a,b] et introduisent des variantes comme la possibilité de dépasser ou de reculer [Jian et al., 2005, Maniccam, 2005], l'utilisation de tailles de corps différentes [Nagai et Nagatani, 2006] ou encore l'intégration de la vitesse. Toutefois, ces modèles sont limités à la simulation de trafic unidirectionnel (*i.e.* les piétons avancent dans le même sens).

D'autres modèles proposent une grille bidimensionnelle comme structure spatiale de l'environnement [Burstedde et al., 2001]. La première dimension correspond à la discrétisation de l'environnement sous forme de cellules. Une matrice 3×3 est créée dans chacune des cellules contenant un piéton. L'ensemble de ces matrices représente la seconde dimension. Chaque matrice définit les probabilités de la prochaine position d'un piéton sur les cases adjacentes à celles où le piéton est situé.

Il existe d'autres types de modèles. Nous pouvons citer les modèles considérant que les piétons peuvent parcourir plusieurs cellules [Blue et Adler, 2001, Dijkstra et Timmermans, 2002, Kirchner et al., 2004], ou encore ceux qui prennent en compte une attractivité sur les cellules [Burstedde et al., 2001, Kirchner et Schadschneider, 2002, Schadschneider, 2001].

Les automates cellulaires offrent une grande rapidité de calculs ainsi qu'une grande facilité d'implémentation. Toutefois, ils limitent les mouvements des piétons, introduisent des cas de blocages (car les individus ne peuvent se déplacer que lorsque la case adjacente est libre) et ne permettent pas le contact entre piétons.

3.5.2.2/ MODÈLES À BASE DE FORCES

Les modèles de comportement de piétons à bases de forces fonctionnent dans un espace continu. Ils calculent un vecteur de direction et une accélération (ou une vitesse) pour définir le prochain mouvement du piéton. Ces modèles permettent, par exemple, de reproduire avec succès des phénomènes de panique [Helbing et al., 2000]. Toutefois, ces modèles ne produisent pas de trajectoires réalistes sur une faible densité d'individus.

[Okazaki et Matsushita, 1991] ont développé un modèle basé sur les forces magnétiques pour analyser le déplacement des piétons en empruntant une équation de mouvement utilisé pour les champs magnétiques. [Helbing et Molnar, 1995] proposent un modèle plus robuste pour le déplacement des piétons, conçu par analogie avec la physique newtonienne. Ce modèle se base sur les trois hypothèses suivantes :
– chaque piéton dispose d'une vitesse de marche qu'il adapte en respectant une valeur de confort ;
– un piéton maintient une certaine distance avec les autres ; et
– un piéton cherche à atteindre un point particulier de l'espace.

Ces trois hypothèses sont ensuite formalisées sous forme d'une combinaison de forces pour définir le prochain mouvement du piéton. Pour déterminer la prochaine position d'un

piéton, une vitesse à un instant t est calculée suivant l'équation de l'accélération :

$$m_i \frac{dv_i}{dt} = m_i \frac{v_i^0(t)e_i^0(t) - v_i(t)}{\tau_i} + \sum_{j(\neq i)} f_{ij} + \sum_W f_{iW} \tag{3.1}$$

La position $r_i(t)$ d'un piéton i à un instant t est ainsi donnée suivant cette vitesse, telle que $v_i(t) = \frac{dr_i}{dt}$. m_i représente la masse d'un piéton i, $\frac{dv_i}{dt}$ son accélération, $v_i^0(t)$ la vitesse qu'il désire atteindre, $v_i(t)$ sa vitesse actuelle, $e_i^0(t)$ la direction dans laquelle il souhaite aller, τ_i un paramètre temporel, f_{ij} et f_{iW} sont les forces d'interactions auquel il est soumis avec respectivement les autres piétons et les murs.

La force d'interaction avec un autre piéton j est donnée par l'équation suivante :

$$f_{ij} = \{A_i exp[(r_{ij} - d_{ij})/B_i] + kg(r_{ij} - d_{ij})\}n_{ij} + Kg(r_{ij} - d_{ij})\Delta v_{ij}^t t_{ij} \tag{3.2}$$

$\{A_i exp[(r_{ij} - d_{ij})/B_i]\}n_{ij}$ est la force d'interaction répulsive où A_i et B_i sont des constantes, d_{ij} la distance entre les centres de gravité des deux piétons, n_{ij} le vecteur normalisé pointant du piéton i au piéton j, r_{ij} la somme de leur rayon. Une force de contact $\{K(r_{ij} - d_{ij})\}n_{ij}$ maintient la cohérence de positionnement des piétons (*i.e.* elle empêche deux piétons de se chevaucher). Une force de friction $k(r_{ij} - d_{ij})\Delta v_{ij}^t t_{ij}$ est prise en compte. $t_{ij} = (-n_{ij}^2, n_{ij}^1$ représente la direction tangentielle, $\Delta v_{ij}^t = (v_j - v_i) \cdot t_{ij}$ la différence des vitesses tangentielles, k et K sont des constantes. La fonction $g(x)$ est égale à x si les piétons se touchent (*i.e.* $d_{ij} \leq r_{ij}$), sinon elle est égale à 0. Par analogie à la force f_{ij}, la force d'interaction entre un piéton i et un mur W est définie telle que :

$$f_{iW} = \{A_i exp[(r_i - d_{iW})/B_i] + kg(r_i - d_{iW})\}n_{iW} + Kg(r_i - d_{iW})(v_i \cdot t_{iW})t_{iW} \tag{3.3}$$

n_{iW} correspond à la direction perpendiculaire au mur W.

Il existe de nombreux modèles basés sur une combinaison de forces. Certains travaux se penchent sur la minimisation de l'énergie dépensée par un piéton [Hoogendoorn, 2004, Hoogendoorn et Bovy, 2003, Nakayama et al., 2007]. D'autres combinent une force motrice et une force décrivant la stratégie d'évitement [Löhner, 2010, Maury et Venel, 2009], tandis que certains s'inspirent des forces centrifuges [Chraibi et al., 2010, Yu et al., 2005].

[Johansson et al., 2007] se sont penchés sur la zone d'interaction entre le piéton et les divers obstacles (autres piétons, murs, *etc.*). Ils proposent l'utilisation d'un champs de forces anisotropiques, où la force de répulsion exercée sur les côtés est moins forte que celle exercée dans le sens de la marche.

Malgré tout, les modèles de simulation de foules à base de forces ont quelques difficultés à résoudre certains problèmes, comme les impasses ou l'arrivé de deux piétons face à face. De plus, ils ne permettent pas de reproduire des trajectoires réalistes sur de faibles densités de piétons [Lemercier, 2012].

L'approche à base de forces donne lieu à l'utilisation d'un modèle d'environnement découplé du comportement des entités. Le modèle d'environnement doit fournir les informations nécessaires à son fonctionnement. Dans le modèle de [Helbing et Molnar, 1995] (vu ci-dessus), ces informations connues d'une entité sont : son centre de gravité, sa masse, sa vitesse et sa direction désirée, les centres de gravité et les vitesses des autres entités, les distances aux autres entités et aux murs, les directions perpendiculaires et tangentielles aux murs.En plus de fournir ces informations, l'environnement doit maintenir la cohérence de la simulation en modifiant la position des piétons.

Il existe de nombreux modèles d'environnements pouvant être couplés aux modèles à base de forces [da Silveira et Musse, 2006b, Farenc, 2001, Galland et al., 2009, Lamarche et Donikian, 2004, Paris et al., 2007, Schelhorn et al., 1999, Shao et Terzopoulos, 2005, Tecchia et al., 2001, Thomas, 1999]. Ces modèles d'environnement permettent de représenter un large panel d'informations et intègrent parfois plusieurs vues. Ainsi, ils peuvent être couplés à de nombreux comportements de piétons. Nous reviendrons plus en détails sur ces modèles dans la section suivante, car ces modèles sont plus souvent associés aux modèles basés sur la vision.

Certains de ces modèles d'environnement adoptent l'approche des champs de potentiels [Galland et al., 2009, Shao et Terzopoulos, 2005, Thrun et Bü, 1996]. Ces derniers se basent sur une grille régulière sur laquelle est projeté l'environnement spatial. Sur cette grille certaines cellules peuvent être interdites d'accès ou fournir des champs de répulsion. Ces champs de répulsion indiquent généralement la proximité d'un objet statique, comme par exemple un mur. Cette approche pose quelques difficultés pour déterminer le pas d'échantillonnage, mais permet de représenter tout les chemins possibles. Le calcul des champs de potentiel permet d'optimiser la simulation. Les obstacles statiques ne changeant pas de place, les vecteurs répulsions qui leur sont associés sont invariants dans le temps.

3.5.2.3/ MODÈLES BASÉS SUR LA VISION

Dans les modèles microscopiques présentés jusqu'à présent, nous remarquons l'importance de la zone d'interaction entre le piéton et le reste de l'environnement (autres piétons, obstacles statiques, *etc.*). [Gibson, 1958] met en évidence l'importance de la perception de l'environnement d'un individu pendant son déplacement.

[Turner et Penn, 2002] proposent un modèle basé sur la perception. Le piéton choisit une destination à atteindre située dans son champ de vision et ce en fonction de son objectif et des différents obstacles. Cette destination est recalculée à chaque pas de temps.

Le modèle de [Fajen et al., 2003] s'intéresse à l'évitement d'obstacles. Il est défini par un système d'équations différentielles en tenant compte de la direction du piéton, sa vitesse, la distance et l'angle des obstacles et de ses objectifs.

Il existe également des modèles sur la navigation de robots qui pourraient être utilisés dans le cadre de déplacement de piétons en fonction de leurs perceptions [Huang et al., 2006]. C'est d'ailleurs du domaine de la robotique que certains modèles s'inspirent.

De nombreux modèles basés sur la vision, comme [Noser et al., 1995, Ondřej et al., 2010, Pellegrini et al., 2009, Shao et Terzopoulos, 2007], se basent sur plusieurs niveaux d'analyses similaires à ceux proposés par [Daamen, 2004] (*cf.* page 58).

Les modèles basés sur la vision sont souvent couplés avec un environnement détaillé [Farenc, 2001, Galland et al., 2009, Shao et Terzopoulos, 2007, Thomas, 1999]. Ce type d'environnement contient l'ensemble des données nécessaires au comportement des piétons (sémantiques, positions et tailles des objets, champs de perception des piétons, *etc.*). Il fournit également les outils nécessaires à la simulation et la préservation de la cohérence de l'environnement (génération des perceptions, application des lois de l'environnement, *etc.*).

Le modèle de simulation de [Shao et Terzopoulos, 2007] est un modèle adoptant de nombreuses approches pour la simulation microscopique de piétons.

Le comportement des piétons est défini sur plusieurs niveaux d'analyses similaires à ceux proposés par [Daamen, 2004] (*cf.* page 58). Un diagramme d'état transition est utilisé pour définir le niveau stratégique, qui définit le plan de comportement des activités du piéton. En fonction de l'activité en cours le piéton peut se diriger vers une destination précise. Le niveau d'analyse tactique dépend de l'algorithme A* qui permet au piéton de déterminer le meilleur chemin à suivre pour arriver à destination. Bien que le niveau tactique donne le chemin à suivre, le niveau opérationnel complète se dernier en fournissant de légère modification dans le déplacement (*e.g.* éviter un obstacle sur le chemin à suivre). Ces modifications dépendent des perceptions des piétons (objets alentours, distance à ces objets, *etc.*) et de la direction qu'ils prennent. Le modèle d'environnement associé aux comportement des piétons fournit les perceptions et modifie la position des piétons lorsqu'ils se déplacent.

La complexité du comportement des agents impliques certain ajouts indispensable au modèle d'environnement, comme un graphe pour la navigation. Le d'environnement de [Shao et Terzopoulos, 2007] se compose d'une carte topologique assimilé à un graphe dont les nœuds représentent les différentes régions du monde et les arcs représentent les passages entre les régions. Chaque région contient une carte pour la perception des entités et une carte pour la navigation.

La carte de perception contient une grille uniforme pour stoker les entités statiques et un arbre spatial (un quadtree) pour stocker les entités dynamiques. La carte pour la navigation est une structure hybride composée d'une grille et d'un arbre spatial (un quadtree). L'arbre spatial décompose l'espace pour accélérer la recherche de chemin. Les cellules de la grille correspondent aux feuilles de cet arbre. Ces deux structures contiennent les informations nécessaires à la planification de chemin (*e.g.* type d'objet, position, type d'occupation, *etc.*). Contrairement à la grille, le quadtree permet de trouver rapidement un chemin potentiellement non optimal ou aucune solution.

La dynamique du modèle consiste à générer les perceptions et mettre à jour les données. Le processus de perceptions est défini pour les piétons sous forme de requêtes. Les piétons peuvent faire quatre types de requête différentes : (i) la détection de la hauteur du sol, (ii) la détection visuelle (percevoir les entités), (iii) les chemins de planification entre les régions et (iv) la localisation d'un objet. Ce modèle est précis mais trop coûteux en ressources de calcul pour un grand ensemble d'entités. De plus, certains choix sont discutables, comme par exemple l'utilisation d'une grille pour stocker les entités statiques ou même le fait que les piétons gèrent eux même les conflits habituellement du ressort de l'environnement. Ce dernier exemple pourrait produire un état incohérent de l'environnement si les piétons ne résolvent pas correctement ces conflits. Pour la grille stockant les entités statiques, il existe d'autres outils ayant un rapport coût mémoire sur la couverture spatiale plus avantageux. [Shagam, 2003] montre l'intérêt de l'utilisation d'un arbre spatial dans le génération de perceptions suivant l'algorithme de "*frustum culling*".

Le modèle de simulation de [Farenc, 2001] propose un modèle d'environnement urbain, dans lequel elle intègre les informations nécessaires à l'animation comportementale et à la simulation d'un système urbain. Le modèle de comportement des piétons considère les niveaux d'analyse tactique et opérationnel. Le comportement du piéton pour la navigation se base sur un comportement prudent (*e.g.* traversé au passage piétons)

et imprudent (*i.e.* traversé sans tenir compte de la signalétique de sécurité routière). Le niveaux tactique consiste à éviter les obstacles sur le parcours.

Le modèle d'environnement proposé est composé de plusieurs modules : la structure du monde (basé sur le modèle de [Thomas, 1999] et de [Donikian, 1997b]), la représentation géométrique des espaces, la perception des piétons, les actions des piétons et l'animation des piétons. La structure du monde sert principalement à la navigation du piéton au sein de l'environnement. Elle se compose d'une structure hiérarchique et d'une structure topologique. La structure hiérarchique est définie suivant le type des composantes urbaines (quartier, rue, *etc.*) et représente le monde sous la forme d'un arbre associé à différents niveaux de représentation [Thomas, 1999]. La structure topologique permet de structurer les informations contenues dans le mondes (*e.g.* les entités, leur type, leur position, le niveau hiérarchique) sous forme d'un graphe de circulation [Donikian, 1997b]. Cette structure relie les feuilles de l'arbre de la structure topologique. Les nœuds du graphe de circulation correspondent à des espaces géométriques. Les nœuds dont les espaces sont connexes sont automatiquement liés par un arc. Cet arc représente le type de frontière entre les espaces. Il permet de distinguer les frontières traversables, non traversables (*e.g.* un mur) et manipulables (*e.g.* une porte). Pour les espaces géométriques dans lesquels les piétons circulent librement, elle propose une décomposition suivant une approche réseau. Le réseau est construit soit suivant un diagramme de Voronoï généralisé, soit suivant un graphe de sommet. Ils donnent les différents chemins pouvant être emprunter par un piéton pour assurer une distance optimale entre les obstacles statiques et pour assurer le plus court chemin pour traverser un espace libre, respectivement. Ce choix est lié au comportement des piétons qui permet de distinguer les piétons "*curieux*" des piétons "*paresseux*". Le processus de perception n'est pas fondé sur un cône de vision, mais directement sur un accès aux informations demandés par le piétons. Et les actions des piétons sont directement mise à jour au sein de ce système. L'évitement de collision permet d'éviter la décohérence de la simulation lorsque les modification sont effectuées.

Le modèle de [Farenc, 2001] est complet et présente des déplacements réalistes pour quelques cas de simulation (comme la circulation d'un petit nombre de piétons au sein des rues). Toutefois, il réduit l'autonomie des piétons, engendrant ainsi des biais de simulation.Cette réduction porte également atteinte à une réutilisation du modèle, car le comportement ne peut gérer des événements imprévus. De plus, ce modèle est inadapté à la simulation à large échelle de piétons au sein d'environnement intérieurs.

Le modèle de [Galland et al., 2009] fournit un modèle d'environnement pouvant être couplé à un modèle de comportement. Pour coupler un comportement de piétons à l'environnement, il se base sur le modèle d'interaction influence/réaction [Ferber et Müller, 1996].Le modèle d'environnement se distingue par 5 principaux modules. Ces modules ont pour tâches : (i) la génération des perceptions des piétons ; (ii) la génération des dynamiques endogènes de l'environnement ; (iii) la résolution des conflits entre les dynamiques endogènes de l'environnement et les actions que souhaitent effectuer les piétons ; (iv) la modification des données stockées par l'environnement (*e.g.* la position des piétons) ; et (v) le transfert des informations entre l'environnement et le comportement des piétons (les actions que souhaitent effectuer les piétons et leurs perceptions). Ces différents modules du modèle d'environnement sont définis plus en détail dans le chapitre 6, page 111. Ce modèle se base sur les différentes approches de modélisation dont certaines ont déjà été définies auparavant. Il est adapté à la simulation de foules, mais

pas pour le large échelle.

Les trois modèles présentés proposent un haut niveau de réalisme. En contrepartie, ils sont très coûteux en terme de ressources de calcul. Il n'existe pas à notre connaissance des modèles de simulation de foules à large échelle au niveau microscopique et en temps réel. Le problème réside dans le fait que plus un modèle est précis, plus il est coûteux en ressources de calcul et inversement. Ainsi, nos travaux sont axés sur le développement de simulation de foules permettant de faire un compromis entre les ressources disponibles et la précision de la simulation. Nos travaux se focalisent sur l'aspect environnemental et non la définition des comportements des agents. Toutefois, nous supposons que les modèles de comportement se basent sur une approche réactive, *i.e.* le comportement est défini sur le niveau opérationnel, car il permet de simuler avec précision les comportements des piétons. Cependant, le modèle d'environnement que nous présenterons dans les sections suivantes est suffisamment modulaire pour y intégrer par la suite les outils de support nécessaires à l'utilisation de comportement plus complexes (basés sur les niveaux stratégique et tactique).

Considérant que les modèles basés sur la vision sont ceux fournissant le plus au niveau de réalisme, nous basons nos travaux sur ces derniers. Ils nous serviront de support pour la simulation microscopique, puis seront étendus pour la simulation multiniveau.

3.6/ CONCLUSION

Ce chapitre aborde différents types de simulations pour ensuite finir sur un état de l'art des simulations de foules. Ainsi, après une brève introduction aux principes de la simulation, nous abordons ceux des systèmes complexes.

Dans le cadre de la simulation des systèmes complexes, et plus spécifiquement celle de la dynamique de populations composées d'individus en interaction, deux grands types de simulations peuvent être distingués en fonction du niveau d'abstraction de leurs modèles : macroscopique et microscopique. Les simulations macroscopiques ont la particularité de pouvoir simuler des systèmes à large échelle avec moins de ressources de calcul que les simulations microscopiques. Toutefois, les approches macroscopiques ne permettent pas de modéliser les impacts d'événements dus aux interactions endogènes du système.

Certains travaux ont tentés d'allier les avantages de ces deux approches, comme ceux basés sur l'approche multiniveau. Cette approche combine différents niveaux d'abstraction. Elle consiste généralement à déterminer le « *meilleur* » niveau d'abstraction à utiliser. Cette approche peut ainsi assurer une certaine précision des résultats de simulation tout en préservant les ressources de calcul. Toutefois, les simulations multiniveaux souffrent encore de difficultés concernant la modélisation des transitions entre les différents niveaux d'abstraction.

Nous avons ensuite abordé, la Simulation Orientée-Agent (SOA) qui se base sur les systèmes multiagents. Comme nous l'avions expliqué au second chapitre, ces systèmes sont adéquates à la modélisation de systèmes complexes comme les simulations de foules. La SOA se réfère aux modèles individu-centrés et fournit un outil permettant de modéliser et de simuler la dynamique de populations composées d'individus interagissant entre eux. Elle est donc adaptée à la simulation microscopique de foule. Durant cette présentation, nous distinguons la population d'individus de l'environnement. L'environnement

représente le monde physique où sont immergés ces individus.

Ce chapitre se termine par un état de l'art des simulations de foules. Après une présentation de modèles macroscopiques, cet état de l'art se focalise sur les approches microscopiques modélisant le déplacement de piétons et leurs interactions.

En conclusion de ce chapitre, nous considérons la classe d'application dédiée à la simulation de foules dans un environnement intérieur de grande taille et avec un grand nombre d'individus simulés. La réalisation de modèles et d'outils supportant ce type d'applications nécessite d'adapter les modèles existants et de proposer de nouvelles approches. Ceci afin de permettre la simulation d'un large ensemble de piétons suivant des contraintes de simulation courantes (*e.g.* la précision de la simulation, les ressources calculatoires disponibles, *etc.*). Les modèles de simulations de foules doivent définir les comportements des individus et l'environnement dans lequel ils évoluent. Les problèmes actuelles des simulations de foules sont les suivants : ils sont trop coûteux, ils ont des problèmes de monté en charge pour un grand nombre d'entités, ils ne s'adaptent pas aux ressources calculatoires disponibles et les environnements sont monolithiques.

Nous considérons que les modèles des comportements des individus font déjà l'objet de nombreuses études. De plus, certains travaux ont commencé à répondre aux problèmes liés aux simulations de foules, mais uniquement pour le comportement de piétons [Razavi et al., 2011a]. Au contraire, les modèles de l'environnement ne sont pas encore suffisamment étudiés. Les environnements sont monolithiques et donc inadaptés à la représentation de plusieurs niveaux d'abstraction. La représentation de plusieurs niveaux permettrait de sélectionner le meilleur à chaque instant de la simulation et ce en fonction de critères comme par exemple la précision des résultats de simulation.

Nous pensons qu'un modèle dédié à la simulation de l'environnement permettrait d'améliorer la qualité et les performances des simulateurs de foules. Le modèle d'environnement, que nous allons décrire dans les deux chapitres suivant, doit permettre la simulation individu-centrée de la foule et doit dynamiquement adapter son niveau d'abstraction aux contraintes de la simulation, pour palier aux problèmes sur les environnements associés aux simulations de foules.

Dans les deux chapitres qui suivent, nous présentons un modèle organisationnel d'un environnement intérieur pour la simulation de foules, et le modèle holonique basé sur ce modèle organisationnel. Enfin, nous concluons cette thèse par l'application de ces deux modèles sur la simulation des passagers dans les halls d'un aéroport.

II

MODÈLES ORGANISATIONNEL ET HOLONIQUE DE L'ENVIRONNEMENT DÉDIÉS À LA SIMULATION MULTINIVEAU

4

MODÈLE ORGANISATIONNEL D'UN ENVIRONNEMENT SITUÉ POUR LA SIMULATION DE FOULES

L'ÉLABORATION d'un modèle pour la Simulation Orientée-Agents (SOA) doit intégrer non seulement la modélisation des agents mais aussi celle de leur environnement. Dans nos travaux, nous modélisons l'environnement dans lequel sont situés les agents (piétons ou groupes de piétons) par un Système MultiAgent (SMA) assurant des missions spécifiques.
L'objectif de ce chapitre est de proposer un modèle organisationnel de l'environnement pour la simulation de foules de piétons. Pour cela, nous nous appuyons sur le métamodèle CRIO.
Après la présentation d'une vue globale d'un modèle de simulation multiniveau de foules et des missions de l'environnement, nous proposons une modélisation de l'environnement et de ses différentes composantes. Puis, avant de conclure ce chapitre, nous définissons le modèle organisationnel de l'environnement qui est la base du modèle holonique présenté dans le chapitre suivant.

Sommaire

4.1/ INTRODUCTION

L'objectif de nos travaux est de fournir un modèle d'environnement intérieur pour la simulation multiniveau de foules. Ce chapitre décrit la solution pour réaliser un modèle de l'environnement.

Nous rappelons, qu'un environnement intérieur est un *environnement physique* [Odell et al., 2002] délimité par des barrières infranchissables (*e.g.* des murs) et disposant de passages situés à ses frontières permettant de gérer les flux entrant et sortant d'entités. Pour concevoir notre modèle de simulation de foules, la modélisation de la dynamique d'une population de piétons doit être clairement distinguée de celle de la structure de l'environnement. Dans ce chapitre, le modèle de l'environnement proposé est multiniveau. Il intègre les différents niveaux d'abstraction classiquement considérés dans un environnement intérieur : bâtiment, étage, salle, *etc*.

Les principales contributions de ce chapitre sont : (i) la définition du modèle organisationnel de l'environnement pour des simulations multiniveaux, et (ii) la définition d'un ensemble d'indicateurs abstraits qui permettent à une simulation de passer dynamiquement d'un niveau à un autre.

La première contribution permet de fournir un modèle pour la simulation de foules qui est un système complexe. L'approche organisationnelle permet de modéliser les systèmes complexes et permet de définir plusieurs niveaux d'abstractions du système. La seconde contribution met en place la dynamique du système modélisé en sélectionnant le meilleur niveau suivant des indicateurs.

Le modèle décrit dans ce chapitre s'intègre dans un projet de plus grande envergure du laboratoire IRTES-SET [1], visant à fournir un environnement logiciel permettant de simuler la dynamique d'une ville et les différents modes de déplacement : piétons, véhicules, bus, vélos, *etc*. Ce chapitre n'aborde que le cas de la simulation de piétons, mais les modèles proposés peuvent être étendus à d'autres types d'entités.

Dans ce contexte, la section 4.2 est consacrée à la description globale du modèle multiniveau utilisé pour la simulation des piétons et de l'environnement. La section 4.3 décrit les missions propres à la dynamique de l'environnement. La section 4.4, quant à elle, détaille les aspects relatifs à la modélisation de l'environnement. La topologie et les missions de l'environnement sont également décrites. Enfin, la section 4.5 est consacrée à la présentation du modèle organisationnel de l'environnement.

4.2/ SIMULATION MULTINIVEAU DE PIÉTONS : UNE VUE GLOBALE

Simuler des foules de piétons en environnement extérieur ou intérieur nécessite la prise en compte du mouvement d'un grand nombre d'entités tout en maintenant un taux de rafraîchissement de l'interface graphique proche du temps réel [Thalmann et Musse, 2007]. Les SMA sont particulièrement adaptés pour simuler de tels types de comportement au niveau le plus précis où chaque piéton peut disposer d'objectifs et de caractéristiques propres. Toutefois, les SMA peuvent utiliser des ressources de calcul importantes. Dès lors, il devient difficile de maintenir les ressources requises par l'interface graphique et

1. Laboratoire Systèmes et Transports, Institut de Recherche sur le Transport, l'Énergie et la Société

l'environnement intérieur simulé. Cette problématique est un exemple typique dans lequel le niveau de complexité des comportements, et de la simulation en général, doit être adapté en fonction des ressources de calcul disponibles.

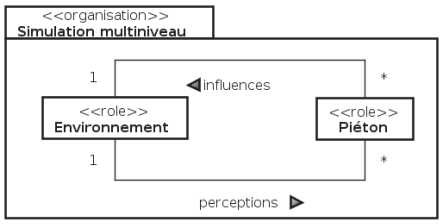


FIGURE 4.1 – L'organisation de gestion des comportements de piétons

Avant d'aborder la modélisation de l'environnement, nous donnons un aperçu global de la simulation multiniveau basée sur le modèle organisationnel de la figure 4.1. Dans ce modèle, nous introduisons deux rôles : `Environnement` et `Piéton`. Le premier fournit à un piéton les moyens de percevoir et de se déplacer dans l'univers simulé. Le rôle `Piéton` est en charge de simuler le comportement d'un piéton unique ou d'un groupe en fonction du niveau d'abstraction considéré. Cette organisation est ensuite combinée au modèle multiniveau pour créer la holarchie d'exécution du système, selon l'approche décrite par [Gaud, 2007].

Différents niveaux d'abstraction peuvent être considérés pour le rôle `Piéton`. Le niveau le plus précis correspond au niveau microscopique : un piéton est associé à un holon atomique. Chaque piéton dispose d'un point de départ dans l'univers simulé (*e.g.* son domicile), et d'un ou plusieurs objectifs à atteindre (*e.g.* son lieu de travail, supermarché, *etc.*). Au niveau supérieur, qualifié de *mésoscopique*, chaque super-holon (holon composé) simule le comportement d'un groupe de piétons. Le groupe est considéré comme un piéton à part entière dont la zone de perception est élargie, et dont la direction de marche correspond à la moyenne de celles de ses membres. Le rôle `Piéton` joué par le super-holon reste le même qu'au niveau inférieur, mais les perceptions et les actions sont agrégées. Le corps ou la représentation du super-holon dans l'univers simulé correspond à l'agrégation des corps de ses membres. Lorsqu'un mouvement est calculé au niveau mésoscopique, le corps des membres du super-holon se déplace selon le mouvement considéré. L'auto-similarité des holons est directement exploitée pour réutiliser le même comportement à différents niveaux d'abstraction.

La figure 4.2 illustre les différents niveaux considérés pour la simulation des piétons. Notons que l'objectif de ce chapitre est d'aboutir aussi à une modélisation de l'environnement qui prend en compte ces différents niveaux. Cette décomposition doit prendre en considération la structure physique et les différentes missions de l'environnement : calculer et fournir ses perceptions à un piéton, gérer les influences/actions des piétons.

Ainsi, dans l'organisation `Simulation multiniveau`, le rôle `Environnement` est en charge de gérer la décomposition structurelle de l'environnement. Ce rôle assure également l'exécution des deux principales missions citées ci-dessus (générer les perceptions, gérer les actions). Le modèle d'interaction entre l'environnement et les piétons permet à ces derniers de percevoir l'état de l'environnement et d'agir en son sein. L'action d'un piéton est généralement modélisée comme la transformation d'un état global [Ferber, 1995], c'est-à-dire la modification directe d'une des variables de l'environnement. Pour suppor-

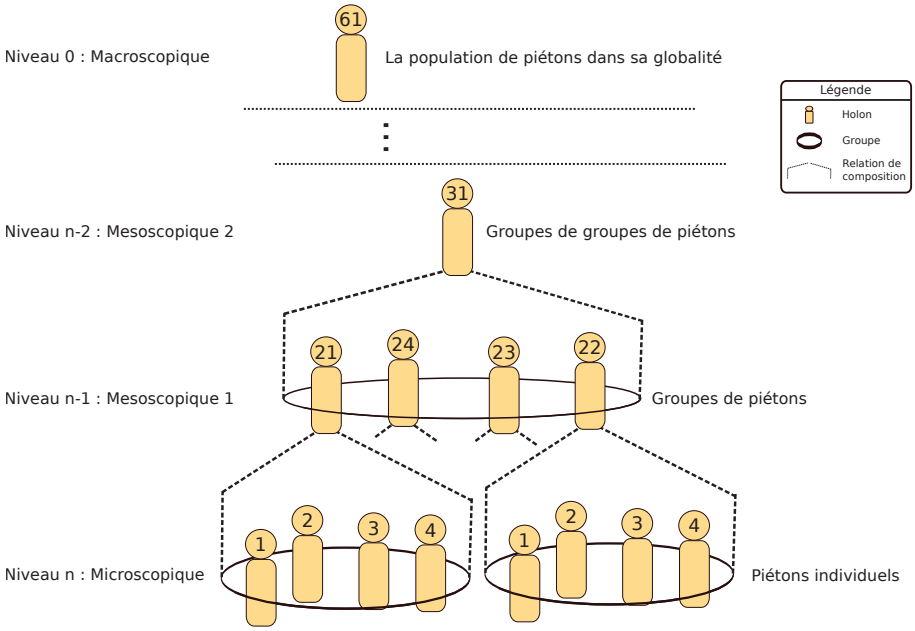


FIGURE 4.2 – Exemple d'une holarchie d'une simulation de piétons

ter cette modification de l'état de l'environnement, nous adoptons le modèle Influence-Réaction [Ferber et Müller, 1996, Michel, 2001, 2007, Weyns et Holvoet, 2004a]. Le cœur du modèle Influence-Réaction consiste à séparer l'action d'un piéton de l'effet qu'elle produit. Dans ce modèle, un piéton produit des influences sur son environnement et non des actions au sens vu précédemment. L'influence représente le désir d'un agent de modifier son environnement d'une certaine façon. Le résultat effectif de cette tentative de modification de l'environnement par un piéton ne peut être calculé sans connaître l'ensemble des influences produites au même instant. Ce modèle se base en fait sur la distinction claire entre deux dynamiques qui sont combinées dans un SMA [Michel, 2007] : (i) La dynamique au niveau de l'agent qui produit des influences. (ii) La dynamique au niveau du système qui calcule la réaction de l'environnement compte tenu de l'ensemble des influences émises simultanément. Pour calculer cette réaction, les influences sont considérées en fonction des lois de l'univers [Ferber et Müller, 1996]. Cette distinction est nécessaire afin de clairement séparer le modèle d'environnement du modèle du système. En effet, l'esprit de l'agent est dépendant du système cible alors que son corps est intégré au modèle d'environnement. La figure 4.3 explicite cette distinction entre l'esprit et le corps d'un agent.

Dans la simulation de piétons présentée dans ce chapitre, le modèle Influence-Réaction a été intégré conjointement avec l'approche organisationnelle pour la modélisation multi-niveau d'un environnement intérieur de bâtiment.

4.3/ MISSIONS DE L'ENVIRONNEMENT

Après avoir donné une vue globale d'une simulation de piétons, nous pouvons constater l'importance de l'environnement dans une simulation de foules. Avant de poursuivre ce chapitre par la présentation de notre modèle d'environnement, nous rappelons ses mis-

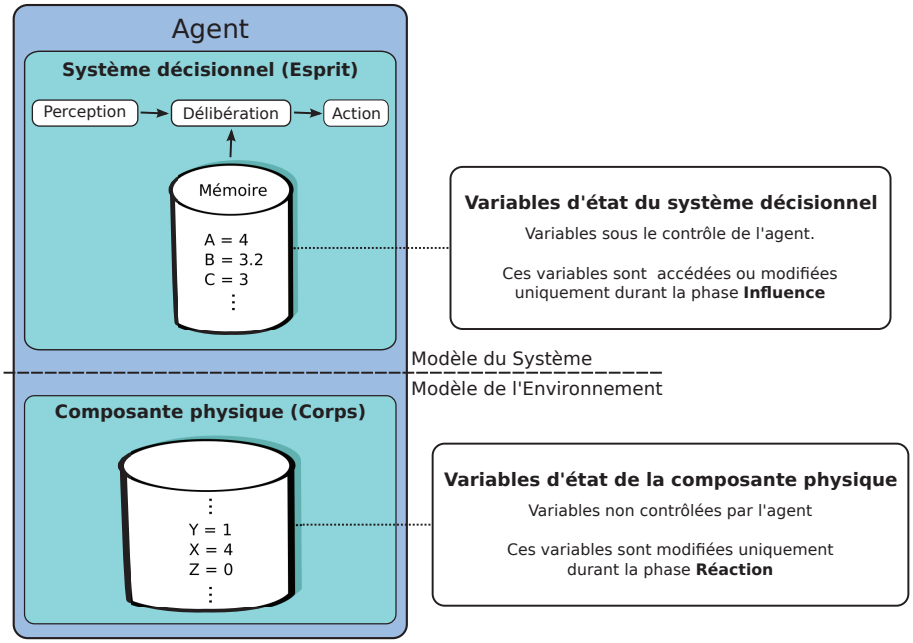


FIGURE 4.3 – Distinction entre l'esprit et le corps d'un agent dans le contexte du principe Influence/Réaction [Michel, 2004]

sions spécifiques dans le cadre d'une SOA. Ces missions sont définies par [Weyns et al., 2005] comme suit :

– **Partager les informations :** L'environnement est une structure commune pour les agents, où chacun d'eux perçoit et agit. Dans le contexte de nos travaux, les données associées sont généralement composées de structures hiérarchiques regroupant tous les objets qui composent le monde simulé et sont utiles aux agents. L'environnement est ainsi hiérarchiquement décomposé en un ensemble de zones, sous-zones et de sorte que chaque zone soit associée à l'ensemble de ses objets. À chaque niveau de cette hiérarchie, les objets sont eux-mêmes reliés à travers un ensemble de structures de graphes pour maintenir l'information topologique, associée à un ensemble donné de la sémantique. Cela donne lieu à une structure de données complexe pouvant être considérée comme une sorte d'hypergraphe. Cette partie de l'environnement contient toutes les structures utilisées pour organiser les informations structurelles, sémantiques et topologiques comme des graphes, octrees, quadtrees, grilles, *etc.*
– **Gérer les actions des agents et les interactions :** Cet aspect est lié à la gestion des actions simultanées et conjointes des agents, et à la préservation de l'intégrité de l'environnement. Par exemple, lorsque deux agents poussent à l'opposer l'un de l'autre la même boîte, l'environnement peut calculer la localisation réelle de la boîte selon les lois de la physique, et les forces appliquées.
– **Générer des perceptions et des observations :** L'environnement doit être localement et partiellement observable. Ainsi, les agents peuvent également gérer l'accès aux informations environnementales et garantir le caractère partiel et local de la perception.
– **Maintenir les dynamiques endogènes :** L'environnement est une entité active. Elle peut avoir ses propres processus indépendamment de celles des agents. Un exemple typique est l'évaporation des phéromones artificielles dans les algorithmes à base de colonies de fourmis.

Ces missions sont exécutées dans un ordre prédéterminé. La figure 4.4 illustre un cycle classique issu de la synthèse entre les cycles proposés par [Ferber et Müller, 1996], et par [Weyns et Holvoet, 2004b]. Cet ordre est garanti par le modèle organisationnel de l'environnement présenté dans la section suivante.

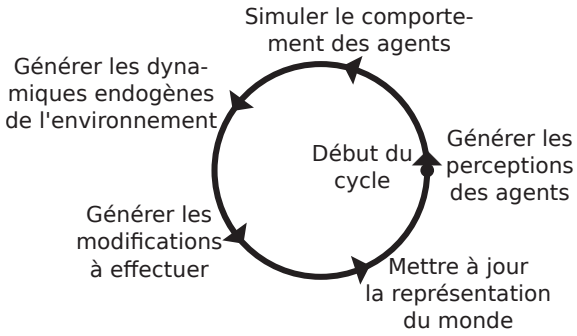


FIGURE 4.4 – Cycle d'exécution d'agents situés et des missions environnementales

4.4/ MODÉLISATION DE L'ENVIRONNEMENT

Cette section présente un ensemble d'abstractions pour la modélisation d'un environnement intérieur.

Selon [Rodríguez, 2005], un système complexe est décrit à l'aide de plusieurs vues. Chacune est modélisée à l'aide de deux éléments principaux : l'analyse structurelle et l'analyse comportementale. La première est concernée par la structure du système ; alors que la seconde se concentre sur son comportement et sur la description des interactions entre ses composants. L'environnement étant lui-même un système complexe [Rodríguez, 2005],il se caractérise par une structure et son comportement. Nous proposons une description de son comportement à l'aide d'organisations. La dynamique de l'environnement est gérée par une hiérarchie d'agents ayant pour objectif de répondre aux missions environnementales. La décomposition structurelle et le comportement de l'environnement sont détaillés dans ce chapitre. La hiérarchie d'agents est présentée au chapitre suivant.

Ainsi, l'une des particularités de l'approche proposée est que la simulation de l'environnement est gérée par un Système MultiAgent Holonique (SMAH), distinct de celui destiné à simuler le comportement des piétons. Le modèle d'environnement que nous proposons a été conçu pour être associé à une grande variété de modèles de systèmes pour simuler les diverses dynamiques d'un bâtiment : simulation d'évacuations, simulation des flux des usagers d'un bâtiment, *etc*.

Pour aboutir au SMAH, il est nécessaire de décrire le modèle organisationnel de l'environnement. Les sections suivantes présentent respectivement la structure de l'environnement et la description de son comportement dynamique à l'aide des concepts organisationnels.

4.4.1/ DÉFINITION DE LA STRUCTURE DE L'ENVIRONNEMENT

Dans cette thèse, l'environnement intérieur est géré par un SMAH en charge d'assurer les différentes missions classiquement assignées à l'environnement dans une SOA. La figure 4.5(a) illustre un exemple simple de décomposition d'un environnement sous forme de zones et la figure 4.5(b) détaille la décomposition hiérarchique correspondante de l'environnement sous forme de holons. Deux super-holons environnementaux H_{21} et H_{22} gèrent ainsi deux zones voisines. Ces super-holons interagissent entre eux pour permettre aux piétons de passer entre leurs zones respectives. Les super-holons génèrent également de manière coordonnée et conjointe les perceptions des piétons qui perçoivent dans leurs zones. La zone H_{21} est à son tour décomposée en trois sous zones, gérées par les holons H_1, H_2, et H_3. Sur la figure 4.5(b) apparaît également le groupe des `Missions environnementales`. Ce groupe a en charge de simuler les missions de l'environnement dans la zone H_3. Les autres holons non décomposés dans la hiérarchie contiennent également ce groupe. Dans ce chapitre, nous n'entrerons pas dans les détails concernant la construction de la holarchie. Nous nous focaliserons sur le modèle organisationnel de l'environnement. Le lecteur pourra se référer au chapitre 5 pour plus de détails sur le modèle holarchique.

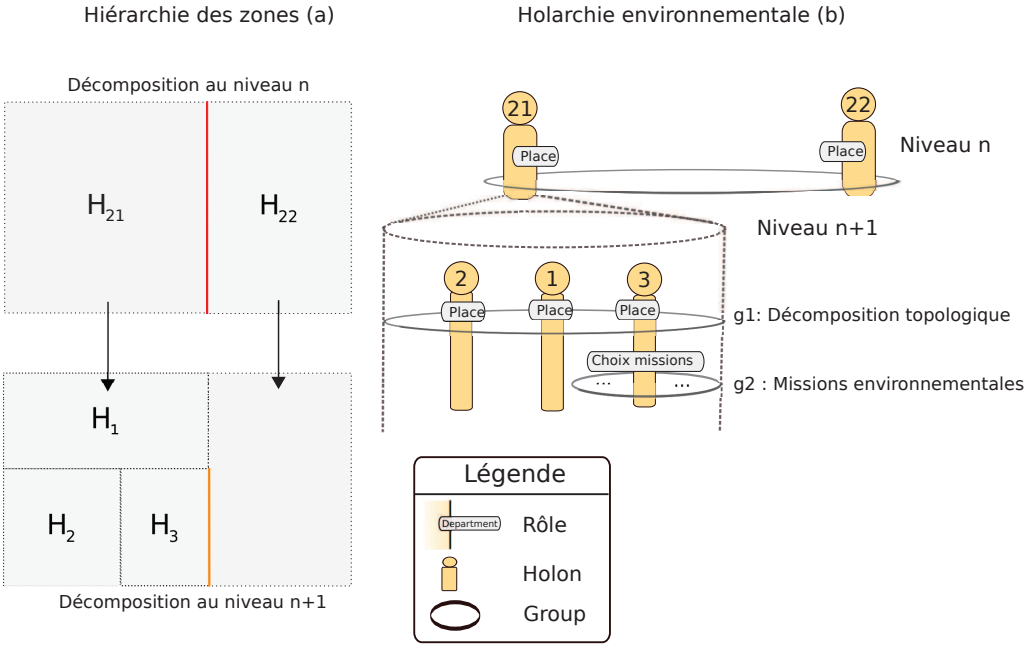


FIGURE 4.5 – Exemple d'une holarchie environmentale

L'environnement est composé d'une structure hiérarchique d'objets et d'un ensemble d'organisations spécifiques en charge des différentes missions environnementales. Sa structure peut se décomposer hiérarchiquement en différentes zones, sous-zones, *etc.*, connectées entre elles. La structure résultante s'apparente à un *"clustered graph"* [Balzer et Deussen, 2007, Eades et Feng, 1996, P. Eades, 1996], qui correspond à un arbre où chaque niveau est un graphe connectant les nœuds du niveau correspondant. Chaque zone est ensuite spécialisée en fonction de son rôle dans le système : bâtiment, étage, couloir, salle, *etc.* La partition utilisée pour décomposer l'environnement est inspirée des travaux de [Pun-Cheng, 2000], [Farenc et al., 1999], et [Thomas, 1999].

Pour assurer les missions spécifiques attribuées à l'environnement, le métamodèle CRIO a été étendu pour intégrer les objets et les organisations nécessaires à la modélisation de ces différents aspects. Cette extension est illustrée par les concepts sur fond bleu dans la figure 4.6. Les concepts ajoutés dans le métamodèle permettent de décrire la structure de l'environnement. Nous introduisons les concepts d'`environnement` et de `zone` afin de décrire cette décomposition structurelle. Nous proposons également de créer des organisations environnementales (et leurs rôles associés) pour décrire le comportement/la dynamique de l'environnement. La suite de cette section détaille les deux décompositions structurelles considérées dans cette thèse : la décomposition statique et la décomposition dynamique.

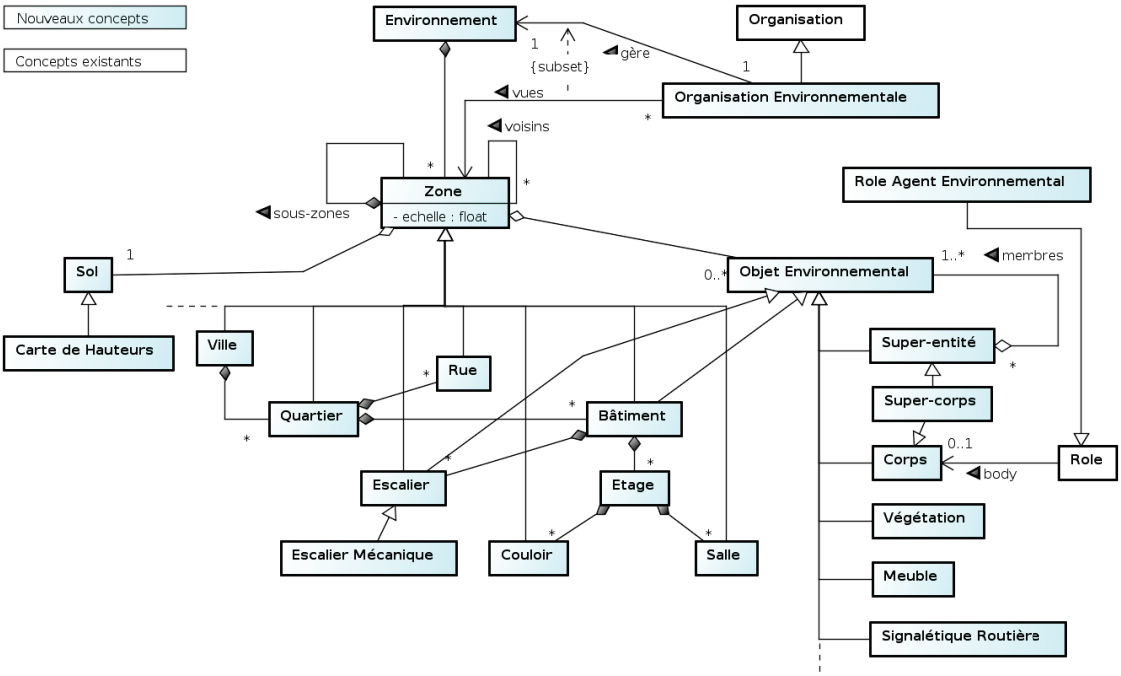


FIGURE 4.6 – Diagramme UML de l'extension du métamodèle CRIO pour la modélisation d'environnements intérieurs

La décomposition statique est souvent réalisée à priori. Elle définie la structure immuable de l'environnement durant l'ensemble du processus de simulation. Plusieurs méthodes de décomposition peuvent être utilisées :

– une décomposition en accord avec le rôle de la zone dans l'environnement urbain : quartier, route, trottoirs [Farenc et al., 1999, Thomas, 1999] ;
– une décomposition en zones géométriquement disjointes : trottoirs, routes [Donikian, 1997b] ;
– une décomposition en places/portails, où une place et un portail représentent respectivement une zone spatiale et un point de passage entre deux places [Luebke et Georges, 1995, Rodrigues et al., 2004].

Dans cette thèse, nous utilisons la seconde approche pour modéliser de grands espaces intérieurs et ouverts. La troisième approche est utilisée pour les parties de l'environnement plus contraintes géométriques : couloirs, salles, cage d'escalier, *etc*. Dans les sous-sections suivantes, nous présentons en détails les concepts permettant de modéliser la

topologie de l'environnement, ainsi que les objets contenus dans cet environnement.

4.4.1.1/ TOPOLOGIE DE L'ENVIRONNEMENT

L'entité principale qui résulte de la décomposition de l'environnement est la zone. Intuitivement, une zone est un espace dont les frontières sont clairement délimitées dans l'environnement. Dans la suite de cette section, nous prenons la zone H_{21} comme illustration (*cf.* figure 4.5).

Chaque zone z est définie telle que :

$$z = \langle D_z, V_z, E_z, g_z, s_z, w_z, e_z, i_z \rangle \tag{4.1}$$

où :
- D_z est l'ensemble des zones décomposant la zone z (exemple : $D_{H_{21}} = \{H_1, H_2, H_3\}$) ;
- V_z est l'ensemble des zones voisines à la zone z. Une zone voisine est une zone possédant au moins une frontière commune avec z (exemple : $V_{H_{21}} = \{H_{22}\}$) ;
- E_z est l'ensemble des objets environnementaux présents dans la zone z. Par exemple, cette ensemble est constitué du mobilier et des corps des personnes se trouvant dans la zone ;
- g_z est la forme géométrique de z, généralement un rectangle ;
- s_z est la description du sol associé à z permettant de déterminer la hauteur de chaque point du sol, s'il n'est pas plat, et la nature de ce sol (carrelage, herbe, *etc.*) ;
- w_z est la constante représentant l'importance de la zone z dans le processus de simulation : plus cette constante est importante, plus la zone sera considérée comme prioritaire durant la simulation. Cet accord de priorités est important pour l'allocation des ressources du simulateur. Cette constante permet au concepteur du scénario de focaliser la simulation sur des zones d'intérêt ; et
- e_z est l'échelle de la zone z (*cf.* les détails ci-dessous). L'échelle permet de donner une classification des différents niveaux d'abstraction du système.

L'échelle d'une zone correspond au rapport de l'approximation entre la représentation des éléments (*e.g.* groupe de piétons, groupe de bancs, *etc.*) que contient la zone et la représentation la plus fine possible de ces éléments (*e.g.* piétons, bancs, *etc.*). Ainsi, l'échelle e_z, d'une zone z, est définie telle que :

$$e_z = \frac{u_z}{a}, \text{ avec } a \in \mathbb{R}_+^* \text{ et } u_z \geq a \tag{4.2}$$

L'unité métrique u_z correspond à l'approximation de l'atome spatial a par rapport à e_z. a est l'unité spatiale atomique représentant le plus petit volume non décomposable présent dans l'environnement. u_z est utilisée par exemple pour mesurer le plus petit mouvement possible à une échelle donnée.

Les propriétés invariantes des zones sont les suivantes :
- **Propriété topologique :** L'union des surfaces de sous-zones est incluse dans la surface occupée par la zone mère. Nous ne posons pas de propriété concernant la couverture totale de ses sous-zones par z afin de ne pas restreindre notre choix dans les politiques de décomposition de l'environnement.
  $\bigcup_{c \in D_z} g_c \subseteq g_z$.

– **Propriété concernant les objets :** Un objet environnemental o ne peut appartenir qu'à une seule zone :
$\forall(v,w) \in Z^2, (o \in E_v \wedge o \in E_w) \Rightarrow a = b$.
– **Propriété associée au modèle multiniveau :** L'échelle d'une sous-zone est inférieure ou égale à l'échelle de la zone mère : $\forall w \in D_z, e_w \leq e_z$.

Une zone possède une description du sol. Le modèle décrivant le sol permet à l'environnement d'appliquer correctement la force de gravité en calculant la position verticale de chaque objet (corps des agents inclus). De plus ce modèle peut donner des informations sur la nature du terrain où se déplacent les agents.

4.4.1.2/ OBJETS DE L'ENVIRONNEMENT

Chaque zone contient un ensemble d'objets de l'environnement. Leurs caractéristiques communes sont : (i) une position dans l'environnement, (ii) une orientation dans le plan ou l'espace, (iii) une forme géométrique, (iv) une description sémantique, et (v) une constante de masse représentant l'intérêt ou l'importance de l'objet dans le scénario de simulation. Cette constante est utilisée dans la cadre de la dynamique multiniveau de la simulation. Cette dynamique correspond au choix du niveau à simuler. Elle sera présentée dans le chapitre suivant.

Les corps des piétons sont considérés comme des objets de l'environnement. En effet, le corps d'un agent n'est pas directement géré par celui-ci. Comme les objets, ils sont soumis aux influences de nombreux agents et pas uniquement à celles de leurs propriétaires. L'ensemble des influences produites simultanément est collecté par l'environnement. Ainsi, un agent qui souhaite se déplacer émet une influence de mouvement. Le corps de l'agent correspondant ne sera effectivement déplacé que si l'influence qu'il a émise ne viole aucune loi environnementale et qu'aucune autre influence ne vient contrarier son action.

Le concept de super-entité est crée pour définir un objet de l'environnement représentant un sous-ensemble fini d'objets de l'environnement. Ce concept sert à définir les groupes d'objets ou de piétons, lors de la génération d'une hiérarchie environnementale.

Prenons l'exemple illustré par la figure 4.2 (page 72) où quatre piétons sont associés au super-holon 21. L'auto-similarité des holons impose que le comportement du super-holon 21 suive le même algorithme de déplacement que les holons 1, 2, 3 et 4. La conséquence directe de ce principe est l'existence d'une représentation dans l'environnement pour le super-holon : le super-corps. Un super-corps est une spécialisation d'une super-entité et d'un corps d'agent.

Le choix du moment de création d'un super-corps est influencé à la fois par les contraintes internes au modèle environnemental (performance de calcul, précision minimale de la simulation de l'environnement, *etc.*) mais aussi par le SMA exécutant les comportements des piétons. En effet, cette dernière influence apparaît comme évidente lorsqu'une holarchie est utilisée pour simuler les comportements des piétons. Lorsque des piétons sont regroupés dans un super-holon, alors le modèle de l'environnement peut également regrouper les corps associés. Les travaux de [Gaud et al., 2007] et [Razavi et al., 2011b] offrent deux exemples concrets de modèles multiniveaux de piétons pouvant influencer directement la constitution des ensembles de super-corps.

4.4.2/ DÉCOMPOSITION DYNAMIQUE DE L'ENVIRONNEMENT

La décomposition dynamique est réalisée durant la simulation et fait partie intégrante de la dynamique du modèle de l'environnement proposée dans ce chapitre. Cette décomposition est souvent introduite dans les modèles de simulation d'environnements physiques afin de prendre en compte la complexité (au sens algorithmique) d'accès aux objets présents dans l'environnement. Contrairement à la décomposition statique, elle n'est pas dirigée par la définition *a priori* d'une topologie de l'environnement. Son objectif est de structurer l'espace en un ensemble de sous-zones spatiales pour déterminer efficacement et rapidement l'ensemble des objets sur lequel appliquer une fonction.

Pour cela, nous adoptons des techniques fréquemment utilisées dans le domaine des jeux vidéos ou des "jeux sérieux". Parmi ces techniques, nous pouvons citer :

1. La décomposition de l'espace par des arbres spatiaux. Plusieurs types d'arbres sont disponibles en fonction du nombre de sous-zones supportées par la structure de l'arbre. Les *arbres* kD (ou kD-tree) forment une famille d'arbres spatiaux qui divisent récursivement l'espace en k sous-espaces supportés par les nœuds fils. La propriété des arbres kD est de choisir les axes de partition parallèles aux axes du repère géométrique du monde ;
2. La décomposition en zones visibles depuis un portail [Luebke et Georges, 1995]. Un portail est un point de passage entre deux espaces. L'objectif de cette décomposition est de calculer une structure hiérarchique des zones visibles depuis le point de passage, comme cela est illustré par la figure 4.7 ;
3. La décomposition de l'espace en zones hexagonales [Macedomia et al., 1995]. ;
4. La décomposition en zones triangulaires [Paris et al., 2007], où l'espace est subdivisé suivant une triangulation de Delaunay contrainte [Kallmann et al., 2003].

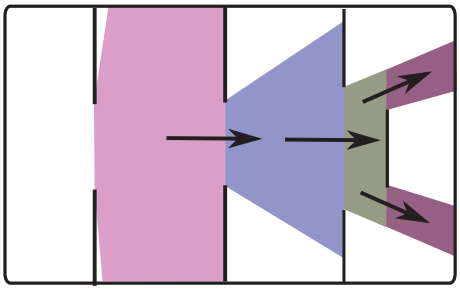
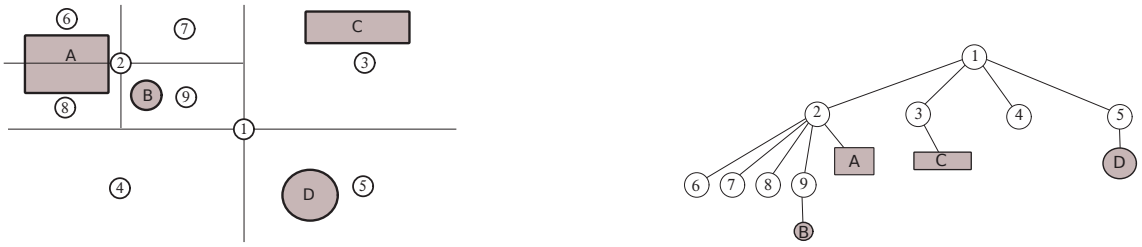


FIGURE 4.7 – Décomposition en zones visibles depuis un portail Luebke et Georges [1995]

La seconde méthode est particulièrement bien adaptée pour les environnements intérieurs, autrement dit, constitués de salles et de couloirs. Car l'espace est dissociable en sous-espaces séparés par de petits portails. Les deux dernières méthodes sont plutôt dédiées à des espaces de grande tailles, impossible à dissocier en sous-espaces en fonction de barrières naturelles. Nous avons choisi la méthode des arbres spatiaux car elle peut être utilisée à la fois dans des espaces intérieurs (*e.g.* un bâtiment) et dans des espaces de grandes tailles.

La figure 4.8 illustre une zone spatiale contenant quatre objets nommés A, B, C et D. La figure 4.8(a) contient une vue du dessus de l'environnement. La figure 4.8(b) contient

l'arbre kD, avec $k = 4$, utilisé par exemple pour décomposer chaque hall de l'aéroport dans le chapitre 6. Les nœuds de l'arbre sont également représentés dans la figure de gauche pour illustrer les sous-zones spatiales qu'ils représentent.



(a) Vue du dessus de l'environnement (b) Structure de l'arbre

FIGURE 4.8 – Exemple de décomposition dynamique d'une zone par un arbre kD-tree, avec $k = 4$

Différents algorithmes ont été développés pour créer et accéder aux éléments d'arbres spatiaux. Les algorithmes spécifiquement développés dans le cadre de cette thèse sont :

- **La mise à jour de l'arbre :** Prenons un ensemble de modifications sur les valeurs des attributs de position d'objets de l'environnement. Cet algorithme recalcule la structure de l'arbre en minimisant les traversées de nœuds et les calculs des attributs associés à chaque nœud. La particularité de cet algorithme est qu'il parcourt l'arbre depuis les nœuds feuilles, plutôt que depuis la racine. Cet algorithme est la base de l'application des déplacements des piétons dans l'environnement.
- **L'algorithme de sélection des objets :** L'objectif de cet algorithme est de calculer le plus rapidement possible un ensemble d'objets dont les géométries sont en intersection avec une forme géométrique donnée en paramètre. Cette algorithme est la base du calcul des perceptions des piétons dans l'environnement. Cet algorithme est donné en annexe B.2.2.1, page 187.

4.5/ ORGANISATIONS DE L'ENVIRONNEMENT

Dans cette section, nous présentons le modèle organisationnel de l'environnement pour la simulation de foules sur la base des concepts définis dans la section 4.4 de ce chapitre.

Les organisations de l'environnement doivent accomplir les différentes missions assignées à l'environnement d'une SOA. Ces différentes missions sont détaillées en début de chapitre : être un vecteur d'interaction entre les agents du système cible, produire des perceptions, gérer les actions des agents, et maintenir la cohérence endogène à l'environnement. Pour distinguer de manière claire les comportements qui composent et participent à la dynamique interne de l'environnement, des comportements qui composent le système cible (la simulation des piétons), un type particulier de rôles a été introduit : les rôles environnementaux (`environmental role`). Ces rôles sont à distinguer des rôles frontières (`boundary role`). Ces derniers sont en effet chargés de gérer les interactions survenant à la frontière entre une application et son environnement, mais font tout de même partie du modèle du système. En revanche, les rôles environnementaux décrivent les rôles qui participent à la satisfaction de besoins purement liés à l'environnement et sont par conséquent considérés comme extérieurs au modèle du système cible.

Deux organisations environnementales principales ont été identifiées et sont décrites

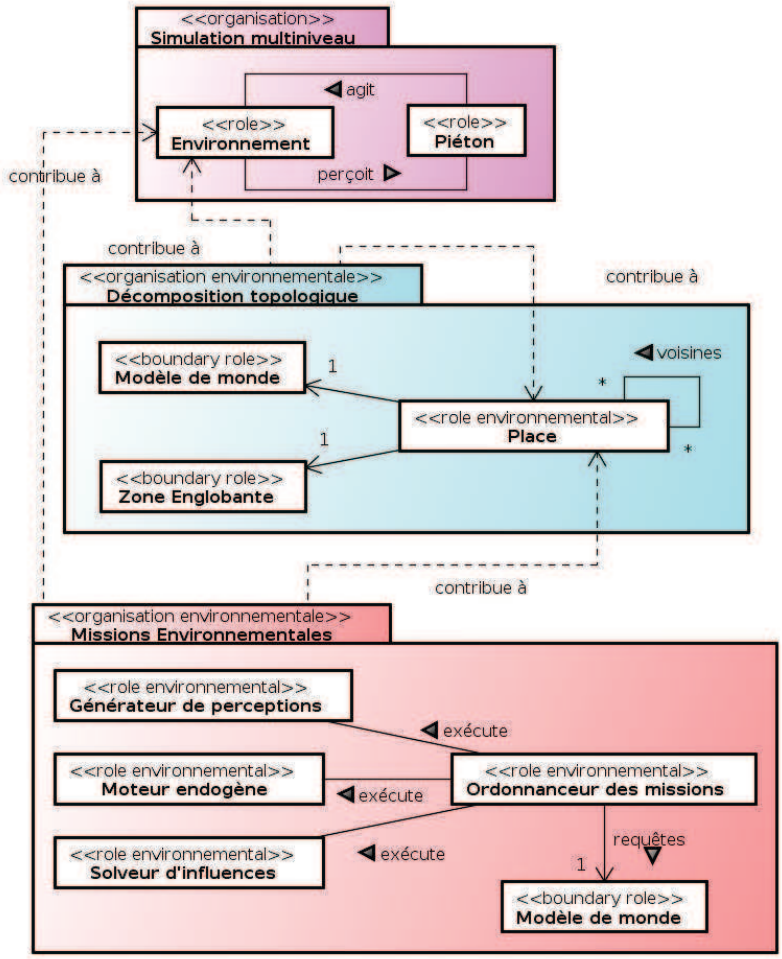


FIGURE 4.9 – Organisations de l'environnement pour une simulation de foules

dans la figure 4.9 : la décomposition topologique et les missions environnementales. L'organisation `Simulation multiniveau` a été décrite dans la section 4.2, page 70.

Comme cela est défini dans la section 2.3.3 (page 28), un rôle peut faire appel à un savoir-faire appartenant à l'agent le jouant. Ce savoir-faire est décrit par le concept de `Capacité` dans le métamodèle CRIO. Le rôle `Environnement` dans l'organisation `Simulation multiniveau` requiert les capacités `computePerceptions` et `applyInfluences`. La première est en charge du calcul de l'ensemble des objets perçus par un piéton. La seconde permet à un piéton d'influencer l'état de l'environnement (au sens du modèle Influence-Réaction). La table 4.1 décrit ces deux capacités. Dans cette table, SM, DT et ME sont respectivement les acronymes des organisations `Simulation Multiniveau`, `Décomposition Topologique` et `Missions Environnementales`. A est l'ensemble des agents piétons. P est l'ensemble des perceptions pour un piéton. I est l'ensemble des influences d'un piéton. Z est l'ensemble des sous-zones de l'organisation. U est l'ensemble des identificateurs.

La capacité permet également, dans le processus de modélisation, d'effectuer l'interface entre deux niveaux d'abstraction adjacents dans la hiérarchie organisationnelle du système [Rodríguez et al., 2007]. Par exemple, la relation *« contribue à »* entre l'orga-

Capacité	*Prototype*	*Requis par*	*Réalisé par*
`computePerceptions`	$A \to \mathcal{P}(P)$	SM:Environnement, DT:Place	DT, ME
`applyInfluence`	$A \times I \to \emptyset$	SM:Environnement, DT:Place	DT, ME
`initializeZoneView`	$Z \to Z$	DT:Modèle de monde	
`initializeIndicators`	$\emptyset \to (U \to \mathbb{R})$	DT:Place	
`getZoneState`	$Z \to Z$	DT:Modèle de monde	
`getIndicators`	$\emptyset \to (U \to \mathbb{R})$	DT:Modèle de monde DT:Zone englobante	
`isExpandable`	$(U \to \mathbb{R}) \to \mathbb{B}$	DT:Place	
`updateEnvironmentTopology`	$Z \times \mathbb{B} \to \emptyset$	DT:Place	
`updateEnvironmentTopology`	$Z \times \mathbb{B} \to Z$	DT:Modèle de monde	
`updateIndicators`	$Z \times (U \to \mathbb{R}) \to \emptyset$	DT:Place DT:Zone englobante	

TABLE 4.1 – Capacités définies dans le modèle organisationnel de l'environnement.

nisation `Missions environnementales` et le rôle `Environnement` (*cf.* figure 4.9) indique que l'organisation fournit une réalisation des capacités requises par le rôle. Ces relations nous permettrons de construire la holarchie de l'environnement et seront détaillées dans les sections suivantes.

4.5.1/ ORGANISATION DE DÉCOMPOSITION TOPOLOGIQUE DE L'ENVIRONNEMENT

L'organisation `Décomposition topologique` représente un niveau dans la hiérarchie de composition de l'environnement. Elle est en charge d'assurer les missions de l'environnement pour une zone à un niveau précis dans la hiérarchie environnementale.

Les objectifs de l'organisation `Décomposition topologique` est de supporter la décomposition dynamique multiniveau de l'environnement et les missions environnementales pour la zone associée. Toutefois, la gestion de ces missions est effectuée par une autre organisation présentée dans la section suivante. Cette organisation fournit donc une réalisation des capacités `computePerceptions` et `applyInfluences`, présentées dans la table 4.1.

L'organisation `Décomposition topologique` est constituée de trois rôles : `Modèle de monde`, `Place englobante` et `Place`.

Modèle de monde : Le rôle `Modèle de monde` maintient la cohérence du Système d'Information (SI) associé à l'ensemble de la zone. Le SI est une base de données dans laquelle l'ensemble des objets de l'environnement sont décrits et stockés. En tant que « *boundary rôle* » , ce rôle `Modèle de monde` gère le SI. Il fournit les services appropriés aux autres rôles de l'organisation : (i) requête sur critères spatiaux, (ii) modification des attributs des objets de l'environnement (position, taille, *etc.*)

Zone englobante : Le rôle `Zone englobante` est un support pour la modélisation multi-

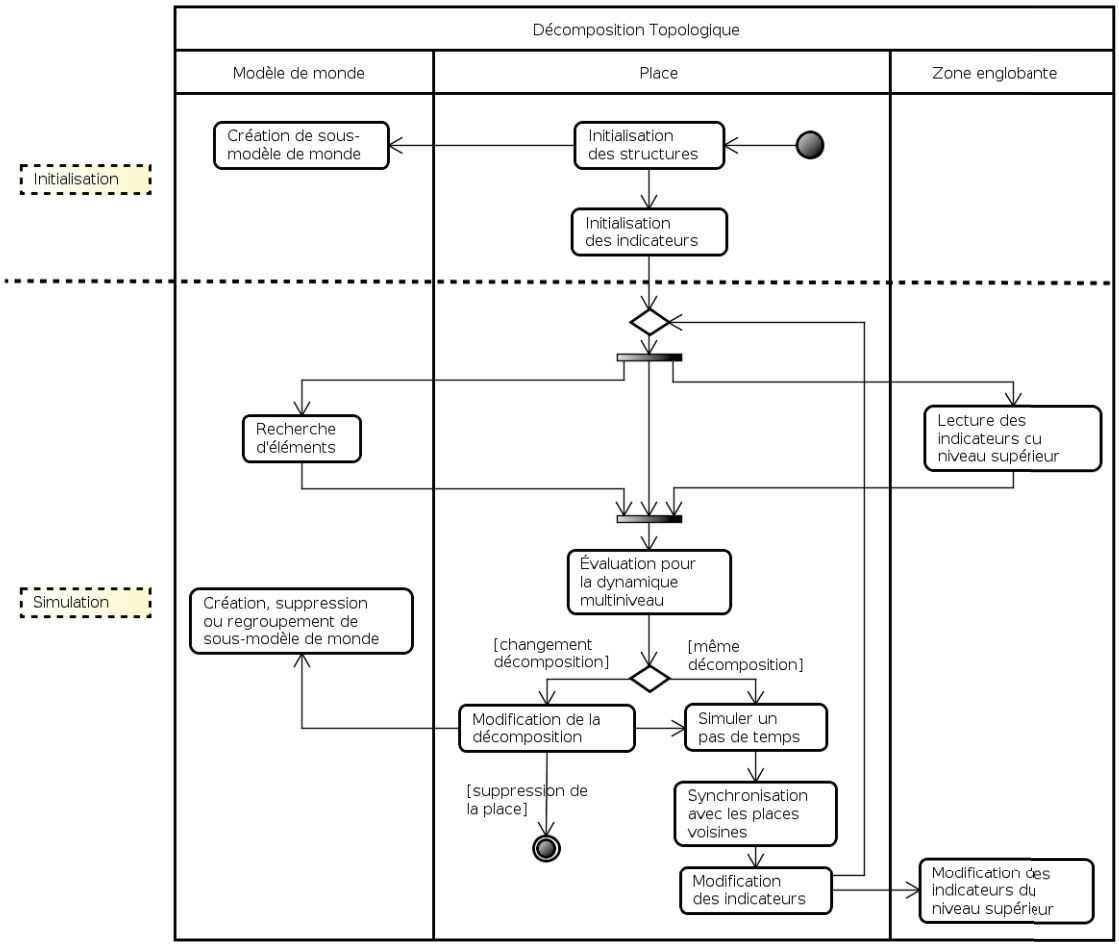


FIGURE 4.10 – Diagramme d'activité décrivant les comportements des rôles dans l'organisation `Décomposition topologique`

niveau de l'environnement. L'organisation `Décomposition topologique` représentant un niveau dans la hiérarchie de composition de l'environnement, il est nécessaire que les places décomposant ce niveau aient accès à un ensemble d'informations dédiées à la dynamique multiniveau. En tant que « *boundary rôle* » , le rôle `Zone englobante` est en charge de fournir aux places l'état de la zone englobante ainsi que les indicateurs et les contraintes donnés par le niveau hiérarchique supérieur. L'ensemble de ces informations seront décrites plus en détails dans la suite de cette section.

Place : Le rôle `Place` décrit le comportement d'une zone particulière de l'environnement. Il a en charge la réalisation des missions de l'environnement pour la zone associée. La `Place` a la capacité de choisir si elle se décompose ou non en fonction de critères et d'indicateurs provenant à la fois du niveau hiérarchique supérieur, mais également de critères et d'indicateurs propres au holon qui jouera ce rôle. Le rôle `Place` est également capable de synchroniser ses activités avec les places voisines dans la hiérarchie structurelle de l'environnement (*cf.* annexe E). Cette synchronisation est rendue indispensable afin de ne pas provoquer de problèmes de causalité entre deux places adjacentes [Bryant, 1979, Chandy et Misra, 1979, 1988, Filloque,

1992, Wang et al., 2005]. Ce genre de problème entraîne des incohérences lors de la simulation, comme par exemple un piéton situé sur deux zones différentes.

Dans cette thèse, nous proposons trois familles principales d'indicateurs :
- **La masse d'une zone** indique l'importance d'une zone de l'environnement dans la simulation. Sa valeur dépend des scénarios de simulation. Par exemple, elle peut être proportionnelle à la densité de piétons dans la zone, ou dépendre de la présence ou l'absence d'un avatar [2] dans cette zone.
- **La profondeur structurelle** décrit la profondeur minimale ou maximale de décomposition d'une zone. Ainsi, il est possible à un rôle `Place` de limiter la profondeur de sa décomposition topologique.
- **La contrainte de ressource** est relative aux limites des ressources disponibles utilisées par une place pour réaliser sa simulation. Cette contrainte permet de prendre en compte des informations bas-niveaux, proche de l'implantation, comme le temps de calcul. Ainsi, il sera possible d'imposer à la simulation d'une place une restriction de son temps de calcul et ainsi de s'approcher d'une exécution temps-réel. Une contrainte de ressource peut également décrire les limites pour tout autre type de ressource bas-niveau (mémoire, bande passante sur un réseau, *etc.*).

Nous considérons que les indicateurs mentionnés ci-dessus peuvent provenir de deux sources : (i) les indicateurs personnels sont propres au holon jouant le rôle `Place` et (ii) les indicateurs et les contraintes partagées sont des informations globales dans le contexte de l'organisation. Tout holon jouant le rôle `Place` peut accéder à ces indicateurs en interagissant avec le rôle `Zone englobante`.

La figure 4.10 présente dans un diagramme d'activités les comportements des rôles de l'organisation `Décomposition topologique`. Le rôle `Place` est central. Les deux autres rôles sont considérés comme des rôles à la frontière de l'organisation et réagissant aux stimulus provoqués par le rôle `Place`.

La suite de cette section est dédiée à la description détaillée du comportement du rôle `Place` durant les deux premières phases de son cycle de vie [3] : (i) son activation/initialisation et (ii) son exécution dans le processus de simulation.

4.5.1.1/ Activation du rôle Place

Lorsqu'il est initialisé par le holon le jouant, le rôle `Place` reçoit une description partielle de la zone spatiale qu'il aura en charge en tant que paramètre d'initialisation. L'annexe C contient les spécifications du format XML permettant de décrire les zones issues de la décomposition statique de l'environnement.

À la demande du rôle `Place`, une vue sur le système d'information est créée par le rôle `Modèle de monde`. Cette vue est limitée à la zone que le rôle `Place` a en charge. Elle permet également de compléter les éléments manquant dans la description de la zone, dans l'équation 4.1 page 77. La création de cette vue est un savoir-faire du holon devant jouer le rôle `Modèle de monde` et est spécifié par la capacité `initializeZoneView` dans la table 4.1.

2. L'avatar est la représentation virtuelle d'un utilisateur réel immergé à travers une plateforme de réalité virtuelle.

3. Le cycle de vie d'un rôle est défini dans CRIO : `http://www.janus-project.org/Organizational_Janus_MetaModel`

Ensuite, le rôle `Place` initialise ses propres indicateurs qui lui permettent de décider de la politique de composition et de décomposition à appliquer durant le processus de simulation. Nous considérons que les types et le nombre des indicateurs sont des propriétés propres à chaque holon jouant ce rôle. Par conséquent, l'initialisation de ces indicateurs est une capacité appartenant au holon, ici `initializeIndicators`.

4.5.1.2/ Comportement du rôle Place

À la fin de son activation, le rôle `Place` réalise une évaluation continue des indicateurs multiniveaux afin de provoquer la modification de la topologie dynamique de sa zone. De plus, le rôle `Place` est en charge des missions environnementales pour cette même zone.

Durant un cycle de simulation, le rôle `Place` réalise les activités suivantes :

1) Il interagit avec les rôles `Modèle de monde` et `Zone englobante` pour obtenir l'état de la zone (ensemble des entités, état de la décomposition topologique, *etc.*) et l'ensemble des indicateurs et des contraintes imposés par le niveau supérieur dans la hiérarchie environnementale.
2) Après obtention des différentes informations provenant des deux autres rôles de l'organisation, le rôle `Place` appelle la capacité `getIndicators` afin d'obtenir les indicateurs propres au holon jouant le rôle `Place`.
3) Afin de déterminer si sa zone peut être décomposée ou non, le rôle `Place` évalue l'ensemble de ses indicateurs en invoquant la capacité `isExpandable` qui lui indique en retour si la zone de l'environnement peut être décomposée.
4) Si la zone de l'environnement doit changer d'état en terme de décomposition topologique, le rôle réalise sa tâche de modification de la topologie en utilisant le savoir-faire de son holon.
5) En parallèle de la modification de la holarchie sous-jacente, le rôle `Place` demande au rôle `Modèle de monde` de mettre à jour sa vue sur sa zone associée.
6) L'étape suivante dans le comportement du rôle est d'exécuter les missions environnementales pour sa zone. Cette activité est un appel aux capacités `applyInfluences` et `computePerceptions`. Les interactions entre l'environnement et les agents piétons sont du ressort de l'organisation `Simulation multiniveau`. La holarchie de l'environnement sera en charge de correctement diffuser les messages d'interactions provenant des piétons au sein de la hiérarchie environnementale.
7) Les influences envoyées par les agents piétons sont des changements d'état de l'environnement. Si ce changement provoque le déplacement d'un objet de l'environnement vers une place voisine, le rôle `Place` source de l'influence, notifie la place voisine en lui fournissant l'influence provenant du piéton. La place voisine l'intégrera dans son processus de simulation en invoquant sa capacité `applyInfluence`.
8) Afin d'éviter tout problème de causalité dans la simulation multiniveau de l'organisation, il est nécessaire que les places se synchronisent entre elles. Les modèles de simulation distribués issus de la littérature fournissent des algorithmes pessimistes [Chandy et Misra, 1979, Ferber et al., 2005, Michel, 2004] et optimistes [Dijkstra et Scholten, 1980, Fujimoto, 1990a, Jefferson, 1985] pour synchroniser les différents holons. L'annexe E (page 243) fournit un aperçu de cette synchronisation détaillée par des schémas issus du processus de conception ASPECS. La synchronisation s'effectue à chaque pas de temps et durant chaque mission environnementale. Il n'est

donc pas nécessaire d'utiliser un protocole complexe pour synchroniser les places, mais seulement d'envoyer les informations nécessaires au maintient de la cohérence du système. Cette synchronisation s'effectue suivant une approche pessimiste avec l'ajout de messages `NULL` [Chandy et Misra, 1979]. Les places attendent les informations qui doivent être échangées (*e.g.* les influences, les perceptions, *etc.*) ou un message `NULL` signalant qu'aucune information ne sera échangée. Les places ne peuvent envoyer les informations à leurs voisines tant que leur fils ne l'ont pas fait.

9) La dernière activité dans le cycle de simulation d'une `Place` est la mise-à-jour des indicateurs dans le holon associé ou dans le niveau supérieur.

4.5.2/ ORGANISATION DES MISSIONS ENVIRONNEMENTALES

L'organisation de gestion des missions environnementales (inspirées de [Weyns et al., 2007]), définit l'ensemble des rôles nécessaires à la satisfaction de l'ensemble des missions de l'environnement pour une zone spécifique. Une instance de cette organisation est intégrée sous forme d'un groupe de production [Gaud, 2007, Rodríguez, 2005] à tous les super-holons jouant le rôle `Place` dans l'organisation décrite dans la section précédente. Ce lien entre les deux organisations est représenté par la relation *« contribue à »* dans la figure 4.9, page 81.

En accord avec le métamodèle CRIO étendu, présenté dans la section 4.4.1, l'organisation `Missions environnementales`, en tant que `Organisation environnementale` est lié à une zone de l'environnement.

Les objectifs de l'organisation `Missions environnementales` sont de : (i) fournir des perceptions aux piétons se trouvant dans la zone associée ou une de ses sous-zones ; (ii) collecter les influences provenant des piétons, et appliquer les réactions environnementales ; (iii) maintenir la dynamique endogène à l'environnement.

En respectant ces objectifs, l'organisation `Missions environnementales` fournit une réalisation des capacités `computePerceptions` et `applyInfluences`, présentées dans la table 4.1 page 82.

L'organisation `Missions environnementales` définit cinq rôles :

Modèle de monde : L'objectif du rôle `Modèle de monde` est de maintenir la cohérence du Système d'Information (SI) associé à la zone. Il est en charge d'appliquer les réactions aux influences provenant des agents. Il fournit également un accès à l'ensemble des objets de l'environnement présents dans la zone.

Générateur de perceptions : Il est en charge de calculer les perceptions des agents pour le prochain pas de simulation.

Solveur d'influences : Toutes les influences émises simultanément dans un pas de temps sont traitées par le rôle `Solveur d'influences` qui calcule les réactions environnementales en accord avec les lois de l'environnement. Il résout les éventuels conflits entre les influences.

Moteur endogène : Le `Moteur endogène` assure la simulation de la dynamique endogène de l'environnement dans la zone. Il est capable d'émettre des influences au même titre qu'un agent applicatif. Ce rôle assure par exemple l'évaporation des phéromones artificielles ou entretient le mouvement d'une balle (non agentifiée) qui a été poussée par un agent et qui continue de rouler. Il assure également le calcul

des variations de variables "globales" (à la zone) de l'environnement telles que la température, la pression ou la gravité.

Ordonnanceur des missions Le rôle `Ordonnanceur des missions` gère l'ordonnancement des missions environnementales. Ce rôle prend en compte les contraintes imposées par une simulation multiniveau en limitant l'ensemble des objets environnementaux pour lesquels une des missions de l'environnement doit être réalisée.

4.6/ CONCLUSION

Nous avons présenté un ensemble d'outils destinés à la modélisation multiagent multiniveau. Cette conception est basée sur deux types de modèles : le modèle du système cible, et le modèle d'environnement.

Les outils de gestion d'une simulation, décrits dans ce chapitre, reposent sur un modèle d'organisation en charge de gérer la décomposition structurelle l'environnement et de simuler ses missions. Cette organisation constitue la base de la conception d'un modèle multiniveau. Elle permet de modéliser la structure organisationnelle d'un simulateur sous forme de groupes et de rôles en accord avec la structure du modèle à exécuter et les différents niveaux d'abstraction qui le composent.

Ce chapitre introduit également un ensemble d'indicateurs de haut-niveau permettant de déterminer si une zone spatiale de l'environnement doit être décomposée ou non. Ils permettent d'évaluer le moment opportun pour effectuer un changement de niveau d'abstraction. Le définition concrète de ces indicateurs est détaillée dans le chapitre suivant.

Les travaux présentés dans ce chapitre représentent une première étape vers la conception d'un simulateur multiniveau. En effet, en plus de notre modèle organisationnel, il est nécessaire de créer un modèle holonique exécutant les rôles au sein d'une hiérarchie holonique. Ce modèle de holarchie est détaillé dans le chapitre suivant.

5

MODÈLE HOLONIQUE D'UN ENVIRONNEMENT SITUÉ POUR LA SIMULATION DE FOULES

UN modèle holonique d'un environnement pour la simulation de foules en environnement intérieur est présenté dans ce chapitre. Le modèle proposé est basé sur le métamodèle CRIO et sur le modèle organisationnel présenté dans le chapitre précédent. Le chapitre courant détaille le Système MultiAgent (SMA) composé de holons en charge de simuler un environnement intérieur. Le modèle multiniveau de l'environnement et de sa dynamique est la principale contribution de ce chapitre. Nous proposons également un ensemble d'indicateurs permettant à chaque holon de l'environnement de déterminer le meilleur niveau d'abstraction à simuler.

Sommaire

5.1/ INTRODUCTION

La simulation multiniveau consiste à moduler dynamiquement le niveau de complexité d'une simulation en fonction de certaines contraintes. Pour y parvenir, nous proposons une approche reposant sur l'adaptation dynamique de la complexité du comportement des composants du système. Nous rappelons que nous focalisons nos travaux sur l'environnement. Ainsi, les composants du système sont ici assimilés aux places composants l'environnement. Les simulations multiniveaux s'adressent avant tout à des systèmes impliquant un grand nombre d'entités en interaction. Son principe vise à faire cohabiter, au sein de la même simulation, les composants du système simulés à des niveaux d'abstraction différents. La même entité peut ainsi changer dynamiquement de niveau de comportement en fonction des contraintes spécifiques telles que la disponibilité des ressources de calcul, par exemple. Pour permettre ce changement dynamique de comportement, il faut être en mesure d'assurer la transition entre plusieurs niveaux de comportements tout en garantissant la cohérence de la simulation.

Ce chapitre présente un modèle de simulation multiniveau d'environnement intérieur pour la simulation de foules. Pour concevoir cette simulation, la modélisation de la dynamique d'une population de piétons doit être clairement distinguée de celle de la structure de l'environnement. En effet, la modélisation structurelle du bâtiment se rapporte à l'aspect environnemental de la simulation, alors que le modèle de la dynamique des piétons concerne davantage le modèle du système. Les environnements intérieurs exhibent naturellement une structure spatiale hiérarchique, voire holonique [Farenc, 2001, Pun-Cheng, 2000, Thalmann et Musse, 2007, Thomas, 1999], qui facilite amplement l'adaptation du modèle proposé. Le modèle de simulation de foules est décomposé en deux sous-modèles, celui pour définir le comportement de foule et celui pour définir la structure du système (*i.e.* l'environnement). Le modèle définissant le comportement de la foule reflète les différents niveaux d'abstraction d'une population de piétons (individu, groupe, foule, *etc.*).

Le modèle de l'environnement proposé intègre les différents niveaux de décomposition structurelle et fonctionnelle de ce dernier. Ces différents modèles (environnement et système) sont ensuite combinés avec les outils de gestion de la simulation multiniveau. Ces outils assurent dynamiquement les transitions entre les différents niveaux d'abstraction, et adaptent automatiquement le niveau de décomposition de l'environnement intérieur.

Ce chapitre est structuré comme suit. La section 5.2.1 fournit la définition des concepts du métamodèle CRIO dédiés à la modélisation de sociétés d'agents. La section 5.2.2 décrit la structure de la (ou des) holarchie(s) de l'environnement. L'ensemble des entités qui composent chaque niveau de la holarchie ainsi que les rôles qu'elles jouent au sein de ces groupes sont également décrits. La structure gouvernementale de chaque superholon ainsi que les règles qui régissent leur dynamique et les mécanismes de prise de décision en leur sein sont décrits dans la section 5.3. Dans le cas de système à forte dynamique, une description de la structure holarchique à des instants clefs de l'évolution de celle-ci est également présentée. La dynamique des holons inclut la description détaillée des indicateurs multiniveaux et de leur usage.

5.2/ MODÉLISATION HOLONIQUE DE L'ENVIRONNEMENT

En s'appuyant sur la modélisation organisationnelle proposée au chapitre précédent, nous construisons un SMA pour la simulation multiniveau. Nous nous intéressons particulièrement à la modélisation de l'environnement. Comme nous l'avons souligné, nous considérons l'environnement comme un SMA disposant de missions spécifiques.

Selon [Rodríguez, 2005], l'environnement peut être modélisé à l'aide de deux éléments principaux : l'analyse structurelle et l'analyse comportementale. La première est concernée par la structure de l'environnement, et plus particulièrement par sa structure hiérarchique dans notre cas ; alors que la seconde se concentre sur le comportement de l'environnement et sur la description des interactions entre ses composants.

L'environnement est décomposé structurellement sous la forme d'une hiérarchie de zones en vue de permettre une simulation multiniveau. Chaque zone est ensuite spécialisée en accord avec son rôle dans l'environnement : bâtiment, étage, couloir, salle, *etc*. Cette première décomposition en zones est réalisée à priori et est immuable durant le processus de simulation. Une seconde décomposition, appelée décomposition dynamique, peut intervenir durant la simulation en fonction des contraintes liées à la simulation et à son simulateur. Cette décomposition est contrôlée par les holons de l'environnement et est dirigée par les valeurs des indicateurs multiniveaux mis en œuvre par ces holons.

Cette section présente un ensemble d'abstractions pour la modélisation holonique d'un environnement intérieur supportant les missions de l'environnement définies dans la section 4.3, page 72. Le métamodèle CRIO présenté dans la section 4.4.1 page 75, est étendu afin d'y intégrer les concepts liés à la modélisation holonique. L'une des particularités de l'approche proposée est que l'environnement intérieur est géré par un SMAH, distinct de celui destiné à simuler les comportements des piétons. Ce modèle d'environnement intérieur a été conçu pour être associé à des modèles de systèmes pour simuler les diverses dynamiques au sein de bâtiments (évacuation de foules, flux de piétons, *etc*).

5.2.1/ MÉTAMODÈLE CRIO ÉTENDU DU DOMAINE DE L'AGENT

Le métamodèle CRIO [Cossentino et al., 2010] distingue trois domaines pour analyser et concevoir un système : le domaine du problème, de l'agent, et de la solution. Ces trois domaines décrivent les concepts correspondant à trois niveaux d'abstraction du processus de modélisation. Les concepts du domaine du problème sont utilisés pour décrire le système en termes organisationnels. Ceux du domaine de l'agent sont le résultat d'un ensemble de transformations des concepts du domaine du problème. Ils sont utilisés pour modéliser une solution orientée-agent du système. Enfin, les concepts du domaine de la solution sont également le résultat d'une transformation et sont dédiés à l'implantation de la solution multiagent.

Dans le chapitre précédent, une extension du métamodèle CRIO du domaine du problème est proposée. L'objectif du présent chapitre est de proposer une extension du métamodèle CRIO pour le domaine de l'agent. Ce métamodèle étendu permet de projeter les organisations et les rôles, définis dans le chapitre précédent, dans une société d'agents. Une partie de cette extension est illustrée par la figure 5.1. Notre contribution [1]

1. Les concepts ajoutés sont représentés sur un fond bleu.

concerne l'intégration des concepts liés à la structure de l'environnement et à sa dynamique (`environnement`, `zone`, `objet environnemental`, *etc.*).

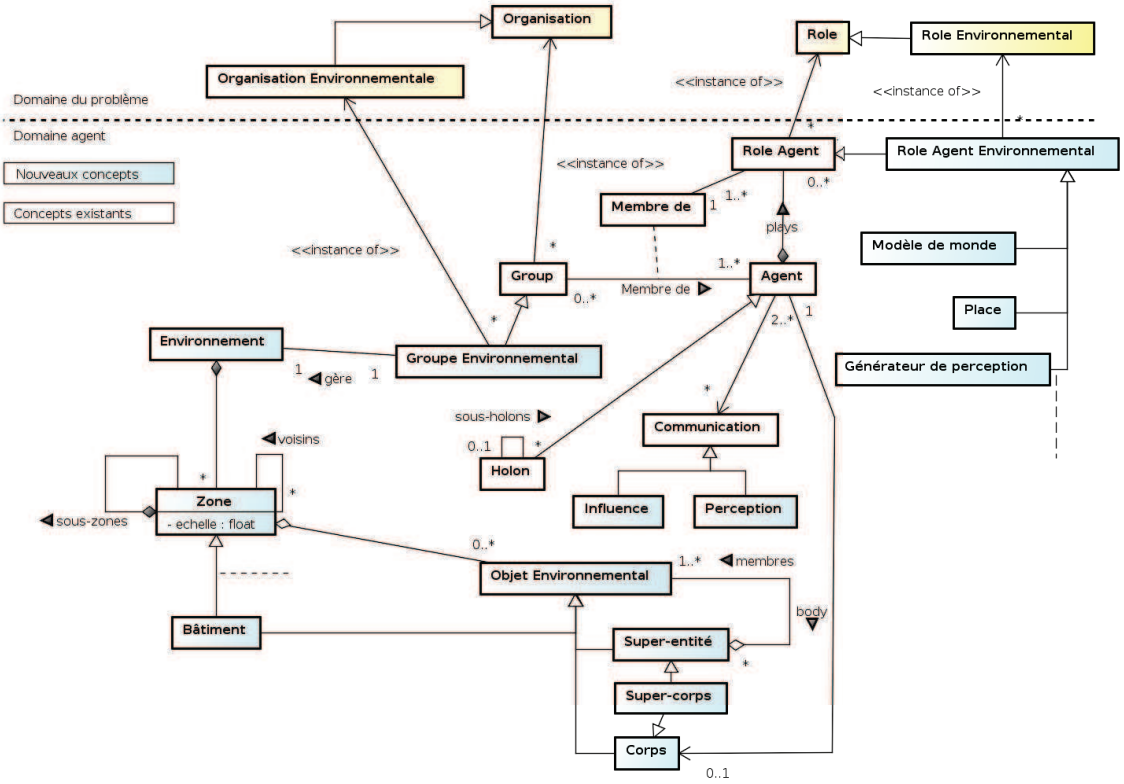


FIGURE 5.1 – Diagramme UML de l'extension du domaine "agent" du métamodèle CRIO pour la modélisation du SMA de l'environnement intérieur

Le domaine du problème correspond au métamodèle CRIO étendu proposé dans le chapitre précédent. Pour une raison de clarté, l'ensemble des concepts ne sont pas reportés dans la figure 5.1.

Toutes les organisations sont instanciées par un groupe d'agents jouant les rôles de l'organisation. Notons qu'un holon est défini comme un agent pouvant être composé d'autres holons. A ce titre, un holon est capable de jouer un rôle dans un groupe d'agents. [Rodríguez et al., 2007] propose un modèle des structures internes régissant les holons. La structure organisationnelle du holon est basée sur deux types de groupes : les groupes de production, dédiés à la résolution du problème ; et un groupe holonique décrivant les interactions entre les sous-holons et le super-holon, indépendamment du problème étudié. Ce dernier permet de décrire comment les messages sont échangés entre les holons membres des groupes de production et le super-holon.

Nous considérons que la capacité de communication entre deux niveaux dans la holarchie est également supportée par le rôle `Zone englobante`, définie dans le modèle organisationnel de l'environnement. Ce choix de modélisation nous permet d'expliciter les interactions multiniveaux entre un sous-holon et son super-holon. Nous considérons également, tout comme [Rodríguez et al., 2007], que le super-holon est capable de créer et de détruire un de ses membres à tout instant.

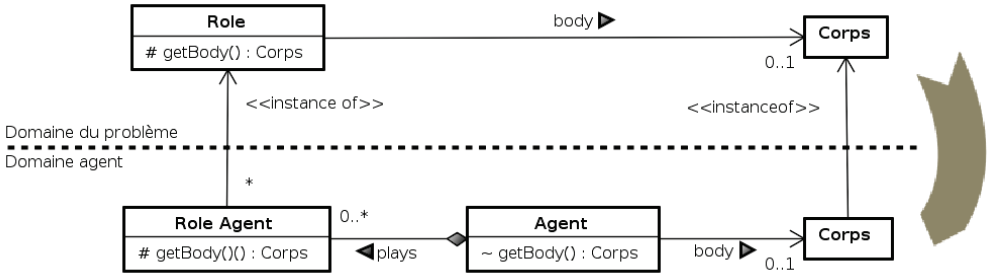


FIGURE 5.2 – Transformation de la relation Rôle/Corps du domaine du problème en relation Role Agent/Agent/Corps dans le domaine de l'agent

L'ensemble des concepts décrivant la structure de l'environnement, à l'exception de l'`environnement` et du `corps`, sont obtenus par une transformation identité entre les deux niveau d'abstraction du métamodèle [MDA, 2003]. Ainsi, les organisations du domaine du problème sont instanciées par des groupes dans le domaine de l'agent ; l'`Environnement` est géré par un groupe environnemental.

De plus, chaque agent possède un corps dans l'environnement. Considérant le principe d'autonomie de l'agent, il est la seule entité réellement autorisée à accéder directement à l'instance de ce corps. La relation *« body »* entre un `Role` et un `Corps` dans le domaine du problème est transformée dans le domaine agent en respectant ce principe. La figure 5.2 illustre cette transformation. Chaque rôle dans le domaine agent (`Role Agent`) accède au `Corps` de l'`Agent` jouant ce rôle en utilisant un accesseur nommé `getBody`. Cet accesseur permet à chaque agent de contrôler et de valider tout accès à son corps.

La section suivante décrit la structure holonique de l'environnement.

5.2.2/ STRUCTURE HOLONIQUE : HOLARCHIE ENVIRONNEMENTALE

À ce stade de la modélisation, l'ensemble des organisations de l'environnement, leurs rôles et leurs interactions est désormais complètement décrit et spécifié. La *conception des holarchies* est la dernière activité de la phase de conception [Cossentino et al., 2010, Gaud, 2007]. Elle est consacrée à l'agentification de la hiérarchie organisationnelle et à la définition des entités en charge de l'exécuter. Son objectif consiste à définir les holons du système et à en déduire la structure de la holarchie. La figure 5.3 contient un exemple d'un environnement contenant les objets environnementaux A, B, C et D. Cet environnement est hiérarchiquement décomposé en zone, numérotées de 1 à 9. Cet environnement simple sert d'illustration dans la suite de cette section.

Pour construire la holarchie de l'environnement, les organisations sont instanciées sous forme de groupes. Un ensemble de holons est ensuite créé à chaque niveau, chacun d'eux jouant un ou plusieurs rôles dans un ou plusieurs groupes du niveau considéré. Les relations de composition entre super-holons et sous-holons sont ensuite spécifiées en accord avec les contributions entre les organisations définies dans la hiérarchie organisationnelle. Par exemple, les super-holons de niveaux n jouent des rôles dans les groupes de niveaux n. Les membres respectifs de ces super-holons jouent des rôles dans les différents groupes de niveaux $n + 1$, qui contribuent au comportement des rôles de

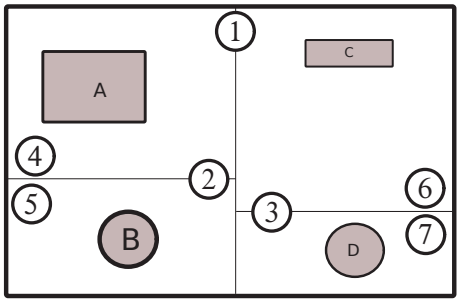


FIGURE 5.3 – Exemple d'un environnement contenant des objets environnementaux et décomposé en zones

niveau n joués par leur super-holon. La hiérarchie organisationnelle est donc directement associée à une hiérarchie de holons (ou holarchie). Tous ces éléments sont synthétisés pour décrire la structure de la *holarchie initiale* du système.

La notation que nous utilisons pour représenter la structure statique d'une holarchie est inspirée de diagramme *« cheese-board »* proposé par [Ferber et al., 2004], et adaptée par [Gaud, 2007] pour mieux représenter les systèmes holoniques. Dans ce type de diagramme, un groupe est représenté par un ovale associé à un nom de la forme *« g[Id]:[NomOrganisation] »* . Les agents sont représentés par des objets en forme de goupilles qui viennent couper les ovales des groupes. Un agent peut ainsi appartenir à plusieurs groupes et donc couper plusieurs ovales. Les rôles sont représentés par des rectangles coupant l'agent et le groupe sur lequel ils sont définis. Les holons composés sont représentés comme les agents à un niveau n donné, englobant l'ensemble de ses organisations internes de niveau $n + 1$ par une ligne pointillée. Chaque agent peut jouer plusieurs rôles au sein d'un même groupe.

Un fragment de la structure finale de la solution holonique pour la modélisation de l'environnement est présenté dans la figure 5.4. La figure 5.4(a) représente les différentes organisations de l'environnement, présentées dans le chapitre précédent. La figure 5.4(b) illustre la holarchie de l'environnement correspondant à la décomposition de l'environnement prise en exemple dans la figure 5.3.

Au niveau 0 de la holarchie, deux holons jouent le rôle `Piéton` et interagissent avec le holon H_1, jouant le rôle `Environnement`, dans le groupe $g0$. Le groupe $g0$ est une instance de l'organisation `Simulation multiniveau`. Dans notre exemple, seul le holon H_1 est décomposé. Il contient une instance de l'organisation `Décomposition topologique` (groupe $g1$). Il contient également une organisation holonique définissant la structure du gouvernement des sous-holons. Ce dernier est inspiré de l'"*apanarchie*", où tous les membres sont impliqués dans le processus de prise de décision (tous les holons sont *Head* au sens de [Rodríguez et al., 2007]). Au niveau 1, le super-holon H_1 joue obligatoirement le rôle `Zone englobante`. En effet, ce rôle a été défini comme une représentation de la zone englobante dans la hiérarchie environnementale. Le super-holon doit par conséquent le jouer pour respecter le modèle organisationnel. Le super-holon H_1 est également en charge de jouer le rôle `Modèle de monde`. Bien que ce rôle puisse être joué par un autre holon dans le groupe $g1$, nous considérons qu'il fait partie des compétences du super-holon.

Les super-holons H_4 et H_5 se décomposent selon les mêmes principes que pour le super-

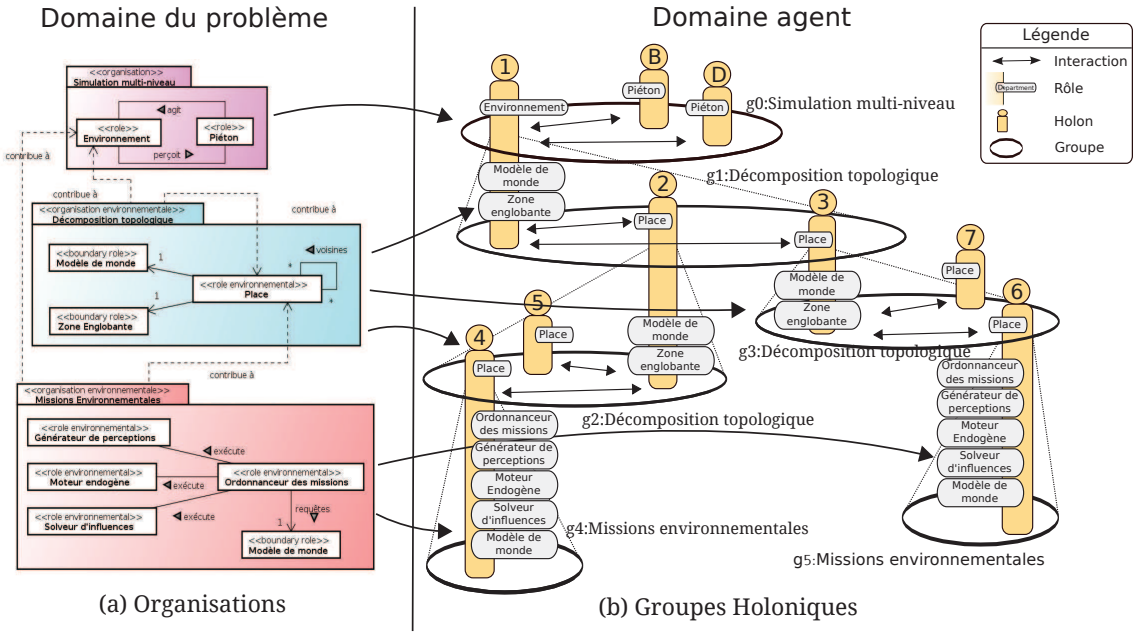


FIGURE 5.4 – Structure holonique du modèle de l'environnement

holon H_1 pour les sous-zones respectives.

Les holons H_6, H_7, H_8 et H_9 sont en charge de zones non décomposées illustrées par la figure 5.3. Ils contiennent chacun une instance de l'organisation `Missions environnementales` afin de calculer les perceptions et gérer les influences des piétons pour la zone associée. Sur la figure 5.4, seule les décompositions des holons H_7 et H_9 sont dessinées. De plus, nous avons arbitrairement décidé que le super-holon dans les groupes $g4$ et $g5$ jouent l'ensemble des rôles de l'organisation `Missions environnementales`. Ce choix n'est pas une contrainte imposée par notre modèle. La section suivante présente les différentes politiques de décomposition envisageable pour constituer la holarchie environnementale.

Les règles qui régissent la construction de la holarchie de l'environnement sont :

– Le rôle `Modèle de monde` a la connaissance de la décomposition structurelle de l'environnement, qu'elle soit définie *a priori* (bâtiments, couloirs, *etc.*) ou durant la simulation en utilisant des structures de décomposition spatiale (arbres spatiaux, *etc.*).
– Il y a une corrélation entre la décomposition structurelle de l'environnement en zones et la décomposition dans la holarchie de l'environnement : chaque zone est associée à un holon possédant le même niveau hiérarchique, et vise et versa.
– Chaque holon dans la holarchie de l'environnement décide de se décomposer ou non. Il utilise pour cela les règles de décomposition données par le rôle `Modèle de monde`. Ce mécanisme de décomposition dynamique est détaillé dans la suite de ce chapitre.
– Chaque holon décomposé jouant le rôle `Environnement` ou le rôle `Place` doit obligatoirement jouer les rôles `Modèle de l'environnement` et `Zone englobante` dans le groupe `Décomposition topologique` de niveau inférieur.
– Les zones non décomposées sont associées à des holons pouvant simuler les missions de l'environnement. Ils jouent les rôles associés à ces missions dans l'organisation `Missions environnementales`.

À la fin de cette étape de modélisation, la structure de la holarchie ainsi que l'architecture des holons qui la composent sont désormais connues. Nous pouvons alors procéder à la définition de la dynamique de la holarchie.

5.3/ MODÉLISATION DE LA DYNAMIQUE HOLONIQUE

Dès lors que différents niveaux de détails sont considérés au sein d'une même simulation, la question de la transition entre ces niveaux devient cruciale. En effet, assurer le passage entre deux niveaux d'abstraction implique que les modèles utilisés par chacun d'entre eux soient compatibles. Cette compatibilité entre niveaux d'abstraction est l'un des problèmes majeurs des approches hybrides, en particulier pour les approches combinant les niveaux microscopique et macroscopique (approche *« micro-macro »*). En effet, dans de telles approches, les deux niveaux d'abstraction considérés sont très éloignés. Il devient alors difficile d'assurer la compatibilité entre leurs modèles. Cette dernière ne peut généralement être assurée que sous certaines conditions strictes : conditions stationnaires, équilibre, *etc*.

L'approche multiniveau tente d'intégrer des niveaux d'abstraction intermédiaires afin de réaliser une transition par paliers entre des niveaux d'échelle très différents, réduisant ainsi les risques d'incompatibilité entre les modèles. L'étude de ces transitions est généralement qualifiée dans la littérature par la notion de *« Pont »* (*"Bridge"*, *"Bridging the gap between"* micro–macro, micro–méso, *etc*.). Ce problème de transition est directement lié à la théorie des systèmes complexes, et touche à des domaines très variés tels que la sociologie [Sawyer, 2003, Schillo et al., 2001, Schwenk, 2004, Troitzsch, 1996], la physique des matériaux [Ghosh, 1986, Li et al., 2005, Lundqvist et al., 2002], l'écologie [Balser et al., 2006, Wu, 1999], l'économie [Dopfer et al., 2004, Kemp et Van den Bergh, 2006, Sato et Namatame, 2001], la robotique [Pettinaro et al., 2003], la biologie [Uhrmacher et al., 2005, Uhrmacher et Priami, 2005], ou encore les systèmes de transport [Hoogendoorn et Bovy, 2001].

Le niveau le plus précis auquel il est possible de simuler un système donné est le niveau microscopique [2] où le comportement des individus et leurs interactions sont modélisés et simulés. Dès lors que sont considérés des niveaux d'abstraction plus élevés que le niveau microscopique (*e.g.* : mésoscopique ou macroscopique), certains aspects du système ne sont plus simulés, et la précision du modèle diminue. Cette notion de précision de la simulation est liée à l'écart entre le modèle simulé et le fonctionnement réel du système. Dans la plupart des cas, les techniques macroscopiques exigent un nombre important d'individus pour fournir une bonne approximation du comportement du système. Par exemple dans le cas de la simulation de piétons ou de trafic de véhicules, l'une des conditions pour utiliser les modèles macroscopiques est de disposer d'une densité forte d'individus (piétons ou véhicules) dans l'espace simulé. Cette contrainte est directement liée à la nature même des approches macroscopiques qui se basent sur des valeurs moyennées des caractéristiques d'une population. Si le nombre d'individus diminue, le modèle s'éloigne du comportement réel du système.

La complexité associée à un niveau d'abstraction dépend des modèles utilisés pour chacun d'eux et de l'application concernée. En revanche, quelle que soit l'application cible,

2. Certains auteurs traitent de la simulation sub-microscopique où les interactions entre les constituants d'une entité sont étudiés [Hoogendoorn et Bovy, 2001, Lamotte et al., 2010].

le problème de transition entre des niveaux d'abstraction différents demeure. La notion d'interaction est au cœur des problématiques de transition entre les niveaux d'abstraction et soulève plus particulièrement deux questions ouvertes et cruciales : Comment évaluer l'impact de l'action d'un individu sur le système et sur les autres individus ? Comment évaluer l'impact du système, de la société ou du groupe sur les actions ou le comportement d'un individu ?

Ces différentes problématiques nous ont amenés à proposer des outils adaptés à la modélisation de l'environnement, permettant d'assurer une transition dynamique entre les niveaux du modèle. Ces transitions sont basées sur l'utilisation d'indicateurs dont certains sont prédéfinis dans notre modèle, et d'autres devront être fournis par le concepteur de la simulation du système cible.

La sous-section suivante est dédiée à la description du modèle dynamique permettant à un holon gérant une zone de se décomposer pour gérer les sous-zones. Un modèle d'évaluation de la cohérence de la simulation multiniveau de l'environnement est donné pour déterminer la meilleure politique de décomposition à utiliser à chaque instant de la simulation.

5.3.1/ DÉCOMPOSITION D'UN HOLON DE L'ENVIRONNEMENT

Chaque holon participant à la holarchie, illustrée par la figure 5.4, doit à tout instant décider s'il doit être décomposé en une holarchie représentant la structure topologique de la zone qui lui est associée, ou s'il doit considérer sa zone comme non décomposée. La seule exception est le super-holon jouant le rôle `Environnement` dans le groupe $g0$. En effet, notre modèle holonique ne possède plus d'intérêt si ce holon ne se décompose pas.

La figure 5.5 illustre la machine à états décrivant le comportement de tous les holons de l'environnement. Les événements `isCollapsable` et `isDecomposable` correspondent à la détection d'un changement de situation à l'aide des indicateurs décrits dans la section suivante. Ils correspondent respectivement aux événements de composition et de décomposition. Les deux algorithmes invoqués sur les transitions sont décrits dans la suite de cette section.

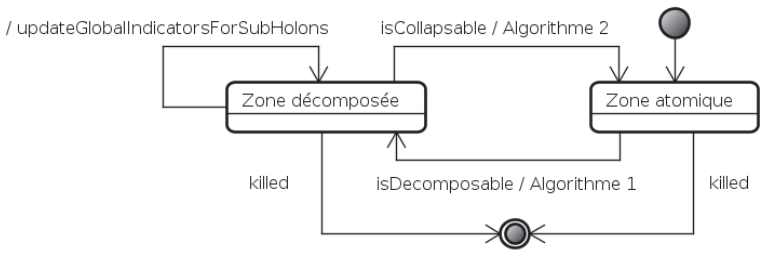


FIGURE 5.5 – Machine à état d'un holon de l'environnement

Nous considérons que les indicateurs peuvent provenir de deux sources : (i) les indicateurs personnels, propres au holon jouant le rôle `Place`, et (ii) les indicateurs et les contraintes globaux dans le contexte du groupe de holons. D'après le modèle organisationnel présenté dans la section 4.5.1 page 82, tout holon jouant le rôle `Place` peut

accéder à ces indicateurs globaux en interagissant avec le holon jouant le rôle `Zone englobante`. Le calcul de ces indicateurs est détaillé dans la sous-section suivante.

Lorsque qu'un holon H décide de décomposer la zone z qui lui est associée, il applique l'algorithme 1. La première étape de l'algorithme consiste à supprimer le groupe de production `Missions environnementales` qui n'est plus nécessaire pour l'exécution des missions environnementales. La seconde étape permet de créer le groupe de production `Décomposition topologique`. Ce groupe contribue également au comportement du super-holon car il fournit les capacités permettant de simuler les missions de l'environnement.

Algorithme 1 Algorithme de décomposition d'un holon associé à une zone de l'environnement

```
1  g := getGroup(H, "Missions environnementales")
2  if g <> nil then
3      releaseRole(H, "Ordonnanceur des missions", g)
4      releaseRole(H, "Generateur de perceptions", g)
5      releaseRole(H, "Moteur endogene", g)
6      releaseRole(H, "Solveur d'influences", g)
7      releaseRole(H, "Modele de monde", g)
8  end if
9
10 S := computeSubZonesOf(z)
11 g := createGroup("Decomposition topologique")
12 requestRole(H, "Modele de monde", g)
13 requestRole(H, "Zone englobante", g)
14 updateGlobalIndicatorsForSubHolons(H,S)
15 foreach zone in S do
16     holon := createHolonIn(H)
17     requestRole(holon, "Place", g, zone)
18 done
19 E_z := ∅
```

L'ensemble des sous-zones de z est calculé en utilisant l'heuristique de découpe associée au holon. L'annexe B, page 175, présente les heuristiques et les algorithmes utilisés pour construire la décomposition topologique de l'environnement. Le chapitre 6 présente un modèle concret de décomposition des zones dédiées à la simulation d'un terminal d'aéroport. Enfin, un groupe `Décomposition topologique` est créé et peuplé par des holons jouant les rôles `Place`. La fonction `updateGlobalIndicatorsForSubHolons` permet de mettre à jour les indicateurs qui seront utilisés par les sous-holons pour leurs propres décisions de décomposition.

Lorsque qu'un holon H décide que sa zone z ne doit plus être décomposée, il applique l'algorithme 2. L'algorithme supprime le groupe de production `Décomposition topologique` qui n'est plus nécessaire pour l'exécution des missions environnementales. Un groupe de l'organisation `Missions environnementales` est alors créé pour permettre au superholon de répondre à ses objectifs principaux : calculer les perceptions des agents et gérer leurs influences.

Ces deux algorithmes permettent de construire, niveau après niveau, le modèle holarchique de l'environnement. Comme l'illustre les comportements décrits dans le chapitre précédent, l'évaluation des indicateurs est réalisée continuellement durant le processus

Algorithme 2 Algorithme de composition d'un holon associé à une zone de l'environnement

```
1  g := getGroup(H, "Decomposition topologique")
2  if g <> nil then
3      foreach holon in getPlayers(g,"Place") do
4          kill (holon)
5      done
6      releaseRole(H, "Zone englobante", g)
7      releaseRole(H, "Modele de monde", g)
8      E_z := createSuperBodies()
9  end if
10
11 g := createGroup("Missions environnementales")
12 requestRole(H, "Ordonnanceur des missions", g)
13 requestRole(H, "Generateur de perceptions", g)
14 requestRole(H, "Moteur endogene", g)
15 requestRole(H, "Solveur d'influences", g)
16 requestRole(H, "Modele de monde", g)
```

de simulation. Ainsi, la holarchie de l'environnement pourra évoluer dynamiquement en étant influencée par les déplacements des piétons et par les ressources allouées à la simulation.

Les algorithmes proposés supposent que le super-holon participant à un groupe de l'organisation `Missions environnementales` possède toutes les capacités nécessaires aux rôles définies dans cette organisation. Chacune de ces capacités correspond à la réalisation d'une des missions de l'environnement. L'annexe B, page 175, détaille les algorithmes associés à chacune de ces missions. Le super-holon propose un service (réalisation d'une capacité selon le métamodèle CRIO) pour simuler ces missions. Une alternative à l'utilisation de services fournis par un seul holon est de définir des sous-holons spécialisés répondant spécifiquement à une mission de l'environnement. Le super-holon joue alors toujours le rôle `Ordonnanceur des missions` mais délègue les autres rôles à ses sous-holons. Ainsi, chaque mission pourrait être implantée nativement (un service) dans chaque holon, ou faire l'objet d'une décomposition organisationnelle. Cette dernière possibilité n'est pas détaillée de cette thèse.

5.3.2/ VERS L'ÉVALUATION DE LA COHÉRENCE D'UNE SIMULATION

Dès lors que l'on considère différents niveaux d'abstraction au sein d'une même simulation, la question de la transition entre ces niveaux devient cruciale. Effectuer une transition entre deux niveaux d'abstraction implique que les modèles utilisés pour chacun d'eux soient compatibles. Cette section introduit quelques outils destinés à faciliter le travail du concepteur d'une simulation et l'élaboration de modèles compatibles. Ces outils permettent également d'évaluer le moment opportun pour effectuer un changement de niveau en fonction des contraintes imposées par le contexte de la simulation.

La précision d'une simulation dépend de l'écart entre les résultats de la simulation et des résultats produits par le système réel. Une échelle permet de classer les différents niveaux d'abstraction du système suivant la précision des résultats de simulations qu'ils fournissent. Le niveau fournissant les résultats de simulation les plus précis est le niveau

microscopique. Nous définissons la précision d'un niveau de la simulation comme étant l'écart entre les résultats de la simulation de ce niveau et ceux sur le niveau le plus précis (le niveau microscopique). Dès lors que l'on considère des niveaux d'abstraction plus élevés que le niveau microscopique, la précision de la simulation diminue. Les niveaux mésoscopiques ou macroscopiques ne constituent qu'une approximation du comportement du système selon un certain point de vue. L'objectif des outils fournis dans cette section vise à estimer le niveau de qualité de cette approximation.

Ce problème peut être apparenté à un problème plus large au sein des SMA qui consiste à évaluer la précision, l'exactitude ou l'efficacité d'un système relativement à la tâche qu'il doit effectuer et aux mécanismes locaux impliqués dans l'accomplissement de cette tâche. Diverses approches ont déjà été proposées dans la littérature, certaines s'inspirant de la biologie (fonction de *"fitness"*), d'autres de la sociologie (fonction d'utilité, satisfaction, *etc.*), ou encore de la physique (entropie). Parmi les solutions inspirées de la physique, l'entropie a été couramment employée, en particulier dans le domaine des SMA réactifs, pour représenter ou tenter de mesurer le désordre (respectivement l'ordre ou l'organisation) au sein d'un système. Diverses méthodes ont été proposées pour calculer l'entropie d'un SMA depuis la notion d'entropie sociale hiérarchique de [Balch, 2000], à l'entropie dynamique et statique de [Parunak et Brueckner, 2001] (et [Parunak et al., 2002]). [Van Aeken, 1999] utilise également l'entropie pour tenter de déterminer le meilleur compromis entre la taille et l'équilibre de sa structure hiérarchique dans les *systèmes multiagents minimaux*.

Même si cette mesure s'avère utile dans de nombreux cas, l'entropie possède deux inconvénients majeurs. Tout d'abord, elle dépend des transformations passées du système : l'entropie ne peut pas être considérée comme une fonction d'état. En cela deux systèmes identiques peuvent être dans le même état, mais avoir des entropies différentes. En second lieu, l'entropie est principalement une mesure globale qui ne tient pas compte des phénomènes locaux du système.

Une solution relativement généraliste réside dans le calcul de l'énergie comme fonction d'état au niveau de l'agent et du système [Contet et al., 2007]. Cette solution est bien adaptée aux systèmes holoniques grâce aux propriétés de composition de l'énergie. Cette énergie permet de tenir compte des phénomènes locaux du système. Cependant, ce type d'énergie n'est pas toujours facilement calculable. D'autres méthodes intéressantes basées sur la physique statistique ou la thermodynamique sont de plus en plus utilisées [Baras et Tan, 2004, Martinas, 2004]. Ces méthodes sont basées sur la fonction de partition Z [3] de laquelle découle toutes les fonctions d'état dans les systèmes thermodynamiques telles que la fonction de Gibbs, l'Énergie libre, l'Enthalpie, l'Enthalpie libre, *etc*. La principale difficulté de ces méthodes concerne les conditions restrictives de son application. En effet, la physique statistique a pour but d'expliquer le comportement et l'évolution de systèmes physiques comportant un grand nombre de particules (systèmes macroscopiques), à partir des caractéristiques de leurs constituants microscopiques (les particules). Pour être en mesure de calculer une valeur significative, le nombre des individus (ici des agents) au sein du système doit être suffisamment important pour être statistiquement significatif.

La formulation de ces indicateurs est bien souvent dépendante de l'application étudiée et donc du modèle du système. Chacun doit déterminer la meilleure analogie face au problème en cours d'étude.

3. Pour plus de précision voir : `http://fr.wikipedia.org/wiki/Fonction_de_partition`

Le principe de base de notre approche réside dans le calcul de l'énergie des holons composant la holarchie de l'environnement. Il s'agit d'une adaptation du modèle proposé par [Gaud, 2007] à la problématique de la simulation d'un environnement. Le niveau de qualité de l'approximation réalisée à un niveau donné de la holarchie est obtenu par comparaison successive des énergies des holons des niveaux inférieurs. L'énergie globale Eg_i d'un holon i donné dans la holarchie selon la définition fournie dans l'équation 5.1. Cette énergie globale caractérise l'état courant du holon i.

$$Eg_i = Ec_i + Epo_i + Epc_i \tag{5.1}$$

Cette énergie se base sur trois autres énergies :
– L'énergie cinétique Ec_i, mesure liée à la dynamique du holon i.
– L'énergie potentielle objectif Epo_i, dépendante de l'objectif du holon i
– L'énergie potentielle de contrainte Epc_i, liée aux éléments qui entravent la progression du holon i vers son objectif (conflits avec les autres holons).

Ces trois mesures peuvent être considérées comme des fonctions d'état, car elles ne dépendent que des paramètres et des caractéristiques courants d'un holon tels que la vitesse, la position relative à l'objectif à atteindre, les positions des obstacles et des autres holons (dans un périmètre d'influence donné). De plus ces indicateurs peuvent être calculés quelque soit le niveau d'abstraction (le niveau dans la holarchie) pris en compte. Que ce soit pour un holon représentant un composant atomique du modèle ou un groupe de composants, la formulation de ces indicateurs énergétiques ne change pas.

5.3.2.1/ INDICATEURS DÉDIÉES À LA SIMULATION MULTINIVEAU

Afin de permettre l'évaluation des trois énergies présentées dans la section précédente, nous proposons trois types d'indicateurs dédiés à la simulation du modèle de l'environnement (*cf.* page 84) :
– **La masse d'une zone** indique l'importance d'une zone de l'environnement dans la simulation. Sa valeur dépend des scénarios de simulation. Par exemple, cette valeur peut être proportionnelle à la densité de piétons dans la zone, ou dépendre de la présence ou l'absence d'un avatar[4] dans la zone. Cet indicateur participe au calcul de l'énergie cinétique du holon Ec_i.
– **La contrainte de ressource** est relative aux limites des ressources disponibles utilisées par une place pour réaliser sa simulation. Cette contrainte permet de prendre en compte des informations bas-niveau, proche de l'implantation, comme le temps de calcul. Ainsi, il sera possible d'imposer à la simulation d'une place une restriction de son temps de calcul et ainsi de s'approcher d'une exécution temps-réel. Une contrainte de ressource peut également décrire les limites pour tout autre type de ressource bas-niveau (mémoire, bande passante sur un réseau, *etc.*). Cet indicateur participe au calcul de l'énergie potentielle de contrainte Epc_i et de l'énergie potentielle objectif Epo_i.
– **La profondeur structurelle** décrit la profondeur minimale ou maximale de décomposition d'une zone. Ainsi, il est possible à un rôle `Place` de limiter la profondeur de sa décomposition topologique.

4. L'avatar est la représentation virtuelle d'un utilisateur réel immergé à travers une plateforme de réalité virtuelle.

La masse de l'objet e (et non la constante de masse w_e de cette objet) décrit l'importance de e à un instant de la simulation. Plus un objet est *« massif »* , plus il participe aux résultats de la simulation et plus il consomme de ressources. Cette masse, notée M_e est définie par l'équation 5.2.

$$M_e = \begin{cases} w_e & \text{si } e \text{ est un objet atomique} \\ \sum_{a \in e} w_a & \text{si } e \text{ est une super-entité} \\ & \text{contenant des objets de l'environnement} \end{cases} \tag{5.2}$$

La masse d'une zone z (et non la constante de masse w_z de cette zone), décrivant l'importance de z durant le processus de simulation, est définie par l'équation 5.3. Plus une zone est *« massive »* , plus elle participe et influence les résultats de la simulation. La masse d'une zone z est proportionnelle à la masse de ses sous-zones et des objets s'y trouvant.

$$M_z = \alpha_z.w_z + \sum_{a \in D_z} \alpha_a.M_a + \sum_{e \in E_z} \alpha_e.w_e \tag{5.3}$$

D_z et E_z sont respectivement les ensembles des sous-zones et des objets de z (*cf.* équation 4.1, page 77). w_z est la constante de masse de z donnée par le concepteur du modèle de simulation pour un cas d'application particulier et représentant l'importance de la zone dans le scénario. w_e est la constante de masse de l'objet de l'environnement e. α_i est une pondération permettant de moduler la contribution de i (z, a, et e) au calcul de la masse M_z. L'ensemble des pondérations est soumis à la contrainte suivante :

$$\sum_{i \in \{z\} \cup D_z \cup E_z} \alpha_i = 1 \tag{5.4}$$

La contrainte de ressource est imposée par le super-holon à ses sous-holons. Son calcul est également basé sur l'utilisation d'un système de pondération et est fonction de la masse des sous-zones. La contrainte de ressources pour un sous-holon a de z est définie par :

$$R_a = (R_z - k_z) \times \frac{M_a}{\sum_{b \in D_z} M_b} \qquad \forall a \in D_z \tag{5.5}$$

k_z est une constante estimant la consommation de ressources par le super-holon pour exécuter ses algorithmes de prise de décision.

5.3.2.2/ MODÈLE DE LA DYNAMIQUE DES HOLONS DE L'ENVIRONNEMENT

À chaque instant de la simulation, les holons participant à la holarchie de l'environnement évaluent les indicateurs décrits précédemment. Cette évaluation permet de déterminer s'ils doivent changer d'état : être un holon gérant une zone décomposée ou une zone atomique.

Comme l'illustre la machine à état présentée dans la figure 5.5, chaque holon est confronté à l'une des deux décisions suivantes :

Cas 1 Si le holon gère une zone atomique, doit-il décomposer cette zone et créer ses sous-holons ?

Cas 2 Si le holon gère une zone composée, doit-il regrouper les sous-zones et détruire les sous-holons gérant ces sous-zones ?

Ces deux cas sont retranscris par l'équation 5.6 décrivant la condition déclenchant le changement d'état du holon (`isDecomposable` devient vrai). Un super-holon doit se décomposer lorsqu'il possède suffisamment de ressources à sa disposition ou que l'évaluation de la cohérence entre les simulations au niveau n et $n+1$ indique que le super-holon n'approxime plus correctement les comportements de ses membres.

$$\begin{aligned}\left[\left(\exists a \in D_z, \left|Eg_z - Eg_a\right| > \epsilon\right) \vee \left(\forall R, R_z \geq \sum_{p \in D_z} \mathrm{g}_R(p) + k_z\right)\right] \wedge \\ (\max_z < i \vee \min_z > i_z) \wedge \\ (E_z \neq \emptyset)\end{aligned} \tag{5.6}$$

Le premier membre de l'équation correspond à la vérification du maintient de la cohérence dans la simulation. Les énergies des sous-holons sont calculées et comparées à l'énergie du super-holon. Si la différence entre deux énergies de deux niveaux différents dépasse la constante d'erreur ϵ, alors le super-holon n'approxime plus correctement l'ensemble de ses membres. Le second membre de cette équation est basée sur l'utilisation de la fonction g_R permettant d'estimer la quantité de ressources R nécessaires pour que les missions de l'environnement soient appliquées sur la zone gérée par le sous-holon p. Cette fonction g_R est dépendante de l'application cible. Chaque super-holon consomme une partie de la ressource R qui lui est allouée afin de calculer les différents indicateurs multiniveaux et de déterminer sa politique de décomposition. La quantité de ressources consommées est donnée par la constante k_z . Les constantes $\min_z$ et $\max_z$ représentent respectivement les profondeurs minimales et maximales dans la décomposition hiérarchique de l'environnement.

Dans le cas 2, le holon est décomposé en un ensemble de sous-holons gérant les sous-zones de z, la zone associée au super-holon. Ce dernier détermine s'il doit conserver ses sous-holons ou les détruire. Ce denier cas de figure correspond à un changement d'état du super-holon. Un super-holon peut supprimer ses membres lorsqu'il ne possède pas suffisamment de ressources à sa disposition pour réaliser la simulation et que l'évaluation de la cohérence entre les simulations aux niveaux n et $n+1$ indique que le super-holon approxime correctement les comportements de ses membres.

$$\left(\forall a \in D_z, \left|Eg_z - Eg_a\right| \leq \epsilon\right) \wedge \left(\forall R, R_z < \sum_{p \in D_z} \mathrm{g}_R(p) + k_z\right) \wedge \min_z < i \tag{5.7}$$

La fonction g_R estime la quantité de ressources R nécessaire pour que les missions de l'environnement soient appliquées dans la zone gérée par le sous-holon p. L'estimation des ressources nécessaires à la réalisation des missions environnementales est une tâche ardue et complexe. Le type de la ressource considérée oriente les définitions de la fonction g_R. Ces ressources sont également dépendante de la nature de l'application considérée.

Si la simulation dispose de toutes les ressources dont elle a besoin, elle est réalisée au niveau le plus précis. Autrement dit, les holons se décomposent jusqu'à fournir une holarchie supportant la simulation microscopique. Les holons du niveau le plus profond de cette holarchie sont toujours exécutés. En revanche, si les ressources viennent à manquer, le simulateur est capable de déterminer les zones qui nécessitent une allocation prioritaire des ressources disponibles. Les indicateurs permettent à chaque super-holon d'identifier quels sont les sous-holons dont le comportement est approximé de manière insatisfaisante. Ainsi, ces indicateurs permettent de déterminer si un sous-holon doit bénéficier en priorité des ressources disponibles.

Les indicateurs sont intégrés au comportement du rôle `Place` par l'intermédiaire d'un service réalisant la capacité `updateIndicators`.

5.4/ CONCLUSION

Nous avons présenté un ensemble d'outils destinés à la simulation multiagent multiniveau dans le cadre des simulations de foules. Cette conception est basée sur deux types de modèles multiniveaux : le modèle du système cible et le modèle d'environnement. Ces deux modèles peuvent être projetés sur deux holarchies distinctes en charge de leur exécution. Ces holarchies offrent le support nécessaire à la mise en place des mécanismes de changement de niveau et à la modulation dynamique de la complexité des comportements d'une simulation. Elles sont indépendantes l'une de l'autre. Dans ce chapitre, seule la holarchie de l'environnement est détaillée.

Les outils de gestion d'une simulation de l'environnement proposés, reposent sur un modèle organisationnel présenté dans le chapitre 4. Ce modèle constitue la base de la conception d'un modèle multiagent holonique, présenté dans ce chapitre. Il permet de modéliser la structure de l'environnement pour la simulation de foules sous forme de groupes de holons.

Ce chapitre introduit également un ensemble d'indicateurs, indépendants de l'application cible, qui permet d'évaluer l'écart entre deux niveaux d'abstraction adjacents. Ces indicateurs tentent ainsi d'apporter une réponse à l'évaluation de la cohérence d'une simulation multiniveau. Autrement dit, ils tentent de définir l'écart entre les résultats de simulation du niveau considéré lors de la simulation et ceux du niveau le plus précis. Ils permettent également d'identifier les éléments de la simulation qui nécessitent en priorité l'attribution des ressources disponibles. Ces outils permettent de déterminer la cohérence de la simulation, ainsi que le coût calculatoire pour la simulation sur chaque niveau. Ainsi, nous pouvons réaliser un compromis entre les contraintes d'exécution de la simulation et la précision des résultats de simulation.

Les travaux présentés dans ce chapitre sont à considérer comme un guide pour la conception de simulations multiniveaux, et nécessitent d'être approfondis et instanciés sur une application particulière. Néanmoins, ils représentent une première étape vers la conception du simulateur. Cette approche a été appliquée avec succès à la simulation de piétons dans un aéroport. Décrite au chapitre suivant, cette application montre comment calculer et utiliser concrètement les indicateurs proposés.

6

SIMULATION D'UN TERMINAL D'AÉROPORT

DANS ce chapitre, nous proposons l'implantation et le déploiement du modèle de l'environnement sur une application particulière : la simulation des flux aux Postes d'Inspection Filtrage (PIF) d'un terminal aéroport. L'objectif principal de cette application est d'évaluer et valider notre modèle de simulation multiniveau. Ce chapitre présente également un ensemble d'expérimentations concernant cette application et les performances de calcul de notre modèle holonique sur les plate-formes JASIM et JANUS.

Sommaire

6.1/ INTRODUCTION

L'objectif de ce chapitre est de présenter l'implantation du modèle de simulation multiniveau décrit dans les chapitres précédents sur un cas d'application concret : un terminal d'aéroport. Un terminal d'aéroport est un lieu permettant le transfert des passagers de leur moyen de transport terrestre jusqu'aux équipements permettant d'embarquer dans des avions.

Bien que la gestion d'un terminal d'aéroport soit une tâche compliquée compte tenu du nombre et de la diversité des acteurs (*e.g.* les voyageurs, les hôtesses, *etc.*), elle peut être facilitée par l'utilisation de logiciels de simulation [Wilson, 2004]. Ces programmes de simulation permettent, entre autre, de déterminer des solutions aux problèmes de congestion en évaluant plusieurs scénarios d'aménagement.

L'application détaillée dans ce chapitre traite de la simulation des flux de voyageurs dans une partie précise de l'aéroport : les Postes d'Inspection Filtrage (PIF) entre les halls d'enregistrement et d'embarquement. Les PIF sont gérés par des opérateurs de sûreté qui effectuent le contrôle des passagers, des équipages, des personnels de l'aéroport, ainsi que leurs bagages de cabine ou effets personnels. L'objectif des voyageurs est de se rendre aux portes d'embarquement pour prendre leur vol. Afin de satisfaire à cet objectif final, ils doivent au préalable enregistrer leurs bagages, puis passer par un PIF.

Dans ce chapitre, la modélisation, l'implantation et l'expérimentation de la simulation d'un terminal d'aéroport dessert deux objectifs : illustrer le bon fonctionnement de notre modèle de simulation sur un cas d'application concret, et évaluer les performances de calcul du simulateur sur les plateformes JASIM et JANUS.

Ce chapitre est structuré comme suit. Dans la section 6.2, un état de l'art succinct sur la simulation des voyageurs au sein d'un terminal d'aéroport est présenté. La modélisation du terminal est présenté dans la section 6.3. Cette section comporte (i) la description du scénario de simulation appliqué et étudié dans ce chapitre, (ii) la modélisation des comportements des voyageurs, et (iii) un modèle concret de l'environnement pour la simulation du terminal. Dans la section 6.4, nous donnons quelques détails sur les choix d'implantation et de déploiement du modèle de simulation. Ce chapitre se termine par deux types d'expérimentations (*cf.* section 6.5) : l'évaluation des tailles des files d'attentes devant les PIF, et l'analyse des coûts de simulation du modèle de l'environnement.

6.2/ SIMULATION DE VOYAGEURS AU SEIN D'UN TERMINAL D'AÉROPORT

Un terminal d'aéroport est un système complexe fait de nombreuses interactions entre les différents acteurs (*e.g.* compagnies aériennes, voyageurs, opérateurs de sûreté, vendeurs, *etc.*). D'après [Wilson, 2004], la modélisation et la simulation offrent les moyens d'évaluer et d'affiner à moindre coût le choix des équipements, des configurations, et des autres facteurs opérationnels. La simulation d'un terminal d'aéroport permet d'étudier ce système sans avoir à installer des équipements coûteux ou à reconfigurer des zones, ce qui pourrait entraîner une perturbation des flux des passagers et des bagages.

Les études portant sur les terminaux d'aéroports sont principalement axées sur l'amélioration de la sécurité [McLay et al., 2009], l'optimisation des flux de voyageurs [Ashford,

1988, Jackou, 2010, Landeghem et Beuselinck, 2002] ou l'analyse des comportements des voyageurs [Bowes, 1999, Ogenyi Omar, 2001].

Les modèles de simulation de foules se classent en deux catégories principales (*cf.* chapitre 2, page 57) : macroscopiques et microscopiques. Dans le cadre d'un terminal d'aéroport, les approches macroscopiques se basent sur les flux de passagers [Curcio et al., 2007, Jackou, 2010], alors que les approches microscopiques se focalisent sur les déplacements individuels des passagers [Joustra et Van Dijk, 2001, Sharma et Otunba, 2012, Szymanezyk et al., 2011].

Les modèles macroscopiques sont souvent plus faciles à implémenter et à instancier. Toutefois, de tels modèles ne peuvent prendre en compte les impacts dus aux interactions endogènes du système. Ainsi, la précision des résultats de ce type de simulation reste limitée. [Jackou, 2010] propose un modèle macroscopique de simulation des flux de voyageurs dans un terminal aéroport (du hall d'enregistrement jusqu'au hall d'embarquement). Il modélise le système par un ensemble d'équations. Ces équations prennent en compte des données statistiques propres au système, comme par exemple : les temps de passage aux différents contrôles, les probabilités de déclenchement de fausses alarmes, *etc*. Le modèle proposé décrit également le processus d'embarquement des passagers, étape par étape. Cette description est utilisée dans ce chapitre pour construire le modèle de comportement des passagers. [Curcio et al., 2007] proposent un modèle de simulation de flux de passagers au sein d'un terminal d'aéroport. Une vérification et une validation du modèle (réalisée selon des données réelles) montre la capacité du modèle de simulation pour reproduire le système réel avec une précision qu'il estime satisfaisante.

En raisons de la pauvreté de la précision des résultats de simulation fournit par les modèles macrospcopique, nous nous intéressons plus particulièrement aux modèles microscopiques. En effet, l'utilisation de tels modèles nous permet d'évaluer les comportements individuels et les interactions entre individus. [Sharma et Otunba, 2012] proposent un modèle pour l'évacuation des passagers d'un aéroport. Les résultats de leur étude ont pour but d'être utilisés pour mesurer l'efficacité des procédures et des protocoles en vigueur. [Joustra et Van Dijk, 2001] se focalisent sur la simulation des voyageurs aux points d'enregistrement pour, entre autre, évaluer la planification opérationnelle des comptoirs d'enregistrement et améliorer l'affectation du personnel. [Szymanezyk et al., 2011] proposent un modèle microscopique de déplacement de piétons au sein d'un environnement. Ils modélisent la configuration d'un terminal d'aéroport à titre d'exemple. Ce modèle général pour la simulation de piétons au sein d'un environnement peut se rapprocher d'une littérature riche dans ce domaine [Dijkstra et Timmermans, 2002, Galland et al., 2009, Jiang et al., 2010, Razavi et al., 2011b, Shao et Terzopoulos, 2005, Thalman et al., 1999, Thomas et Donikian, 2000, Turner et Penn, 2002]. Ces modèles sont utilisés comme outils d'aide à la décision. C'est dans cette catégorie de modèles que se positionne notre proposition.

Notons qu'il existent des études analysant les comportements des passagers en observant le système réel. [Bowes, 1999, Ogenyi Omar, 2001]. Généralement, elles tentent de déterminer les éléments influençant l'évolution du système. Par exemple, [Bowes, 1999] considère que les voyageurs possèdent du *« temps libre »* lorsqu'ils sont dans l'aéroport. Ce temps libre, appelé temps de disponibilité, correspond aux moments durant lesquels un voyageur ne réalise pas une tâche directement liée au processus d'embarquement dans un avion (*e.g.* l'enregistrement des bagages, le passage à un poste d'inspection filtrage, *etc.*). [Bowes, 1999] étudie la relation entre le temps de disponibilité et l'activité

des magasins dans les halls de l'aéroport. Ces études se basent sur des données réelles. Les résultats de ce type d'études peuvent être utilisées pour affiner le comportement des passagers et ainsi augmenter la précision des résultats de la simulation.

6.3/ MODÉLISATION DE L'APPLICATION

Dans le cadre de simulation de déplacements de voyageurs au sein d'un terminal d'aéroport, nous distinguons le comportement des passagers de celui de l'environnement du terminal. Ainsi, pour modéliser l'application, il nous faut définir (i) un modèle de comportement pour les voyageurs (*cf.* section 6.3.2) et (ii) un modèle concret de l'environnement pour la simulation du terminal d'aéroport (*cf.* section 6.3.3).

Toutefois, la modélisation des voyageurs et de l'environnement ne suffit pas. Il faut également définir le scénario utilisé pour la simulation des passagers dans l'aéroport. Nous commencerons donc par aborder ce scénario dans la section qui suit.

6.3.1/ DÉFINITION DU SCÉNARIO DE SIMULATION

L'une des premières étapes dans un projet de simulation est la définition des objectifs et des scénarios de simulation. Dans notre cas, nous cherchons à connaître le nombre de voyageurs en attente aux postes de contrôle.

Nous avons choisi un terminal d'aéroport constitué de trois halls, comme le montre la figure 6.1. Le hall d'enregistrement est situé en bas ; les halls d'embarquement sont situés en haut ; les zones contenant les postes d'inspection filtrage (PIF) sont les deux rectangles gris superposés entre le hall d'enregistrement et chacun des halls d'embarquement ; les portes d'entrée et d'embarquement sont les rectangles fins situés respectivement en bas et en haut ; les blocs grisés sont des zones inaccessibles.

Le scénario de simulation aux PIF est inspiré des travaux de [Jackou, 2010]. Celui-ci définit la configuration de ces postes et la séquence d'actions réalisées par les passagers pour passer les contrôles. Toutefois, notre configuration diffère quelques peu. Les voyageurs attendent dans une file avant d'aller à l'un des PIF. Ensuite, ils sont dirigés un par un, et dans l'ordre, vers l'un des PIF où des contrôles sont réalisés par le personnel de l'aéroport. Pour des raisons de simplification, les PIF allouent un temps approximatif pour le contrôle de chaque voyageur.

Le temps de contrôle en sortie du PIF est compris entre 10 et 45 secondes. Ces bornes sont données par l'étude et l'optimisation de postes de contrôles proposées par [Jackou, 2010] sur le temps de passage aux contrôles, qui est de 10 ou 15 secondes par contrôle. Nous considérons que nous avons au minimum un contrôle obligatoire et que trois autres contrôles optionnels (deux de 10 secondes et un de 15 secondes) peuvent avoir lieu. Le contrôle obligatoire correspond au passage sous le portique. Le terminal contient trois PIF sur chacune des zones de contrôle situées entre le hall d'enregistrement et chacun des halls d'embarquement.

Nous considérons que le terminal contient 2000 passagers placés aléatoirement dans les zones illustrées par la figure 6.1. Les passagers entrent dans le terminal en suivant une loi de génération constante de 1500 personnes par heure simulée. Cette loi est utilisée pendant une heure de temps simulée. Le nombre de voyageurs allant à la zone d'em-

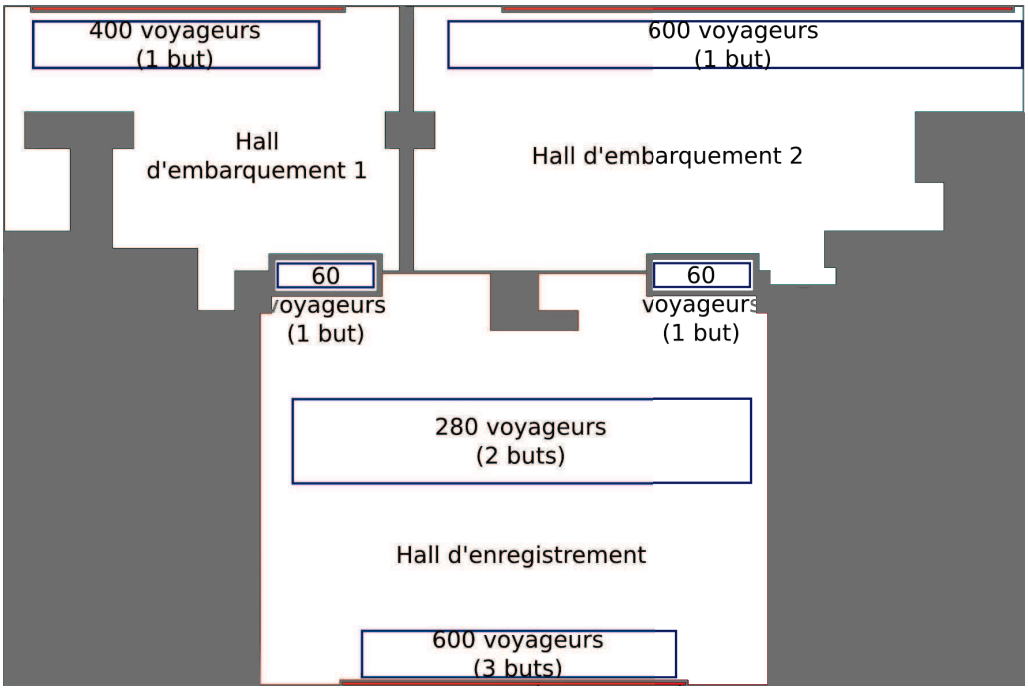


FIGURE 6.1 – Représentation de l'environnement du terminal d'aéroport composé de trois halls et de deux zones de contrôle et de filtrage

barquement de droite (respectivement de gauche) est de l'ordre du tiers (respectivement des deux tiers) du nombre total de voyageurs.

La description complète et détaillée du fichier de configuration décrivant l'initialisation du scénario est donnée en annexe D.

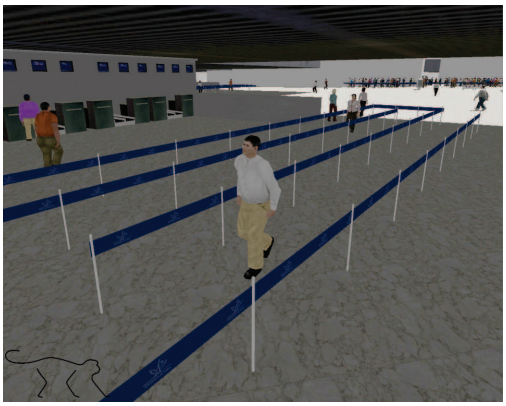
La figure 6.2 présente deux captures d'écrans de l'outil de simulation du terminal de l'aéroport. Cet outil a été réalisé conjointement avec la société Voxelia S.A.S [1] avec laquelle notre équipe de recherche collabore dans le cadre d'un contrat de transfert de technologie. Ces images ont été extraites en rejouant les déplacements des voyageurs produits par notre simulateur. Ces deux captures d'écrans illustrent la topologie des files d'attentes et des postes de contrôles que nous considérons dans le scénario de simulation.

Dans les deux sections suivantes, nous présentons les deux modèles utilisés pour simuler le scénario présenté ici. Le premier modèle concerne la simulation des comportements des voyageurs. Le second modèle est lié à la simulation de l'environnement interne (*i.e.* les halls d'enregistrement et d'embarquement) du terminal.

6.3.2/ MODÉLISATION DU COMPORTEMENT DES VOYAGEURS

Les voyageurs adoptent un comportement semi-réactif (ou hybride), que l'on peut décrire en considérant deux couches sur les trois proposées par [Daamen, 2004] (*cf.* chapitre 2, page 58). Nous n'utilisons pas la couche tactique pour la recherche de chemin en fonction de la topologie du lieu, car cette dernière est suffisamment triviale pour éviter l'utilisation d'un comportement tactique.

1. http://www.voxelia.com



(a) Vue d'une file d'attente devant un guichet d'enregistrement. La file d'attente pour atteindre les postes d'inspection filtrage est visible en fond d'image

(b) Vue d'un poste d'inspection filtrage

FIGURE 6.2 – Captures d'écran de la simulation du terminal de l'aéroport

La première couche permet au voyageur de déterminer la prochaine action à effectuée dans la séquence des actions suivantes : enregistrer son vol, passer à un point de contrôle et embarquer. Cette couche représente le niveau stratégique du processus de décision d'un piéton défini par [Daamen, 2004]. Ces trois étapes constituent les principales phases que nous retenons dans le processus d'embarquement d'un voyageur [Jackou, 2010]. L'agent simulant le comportement d'un voyageur détermine à chaque instant si sa position courante correspond à la position de son prochain objectif. Si c'est le cas, il passe à l'objectif suivant dans la liste ci-dessus. Lorsque l'agent à atteint tous ses objectifs, il disparaît de la simulation. La figure 6.3 présente un diagramme d'état-transition correspondant au comportement de l'agent dans cette couche. La transaction au bureau d'enregistrement et le contrôle en sortie du PIF sont modélisés par une attente de l'agent.

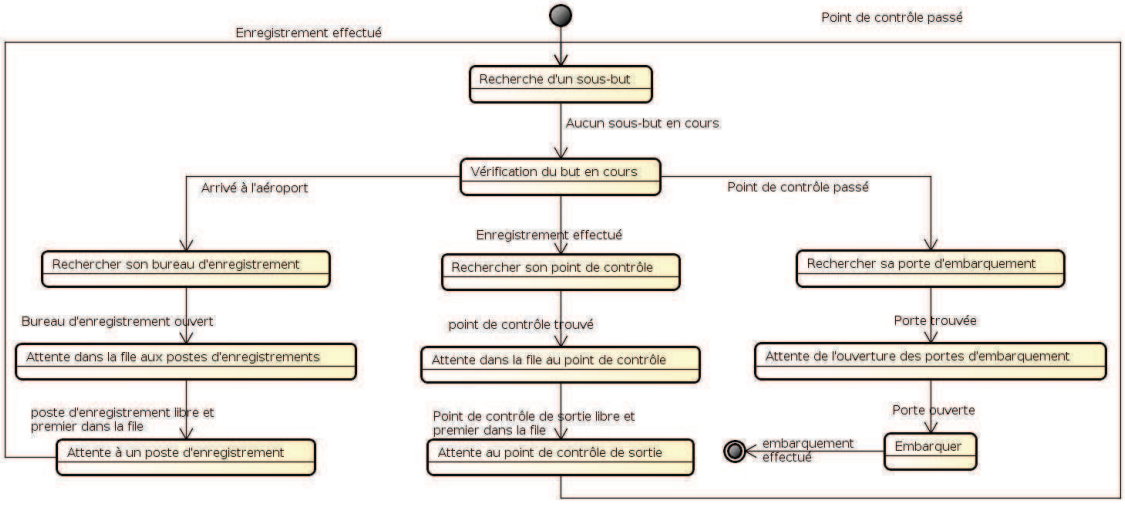


FIGURE 6.3 – Diagramme d'états du comportement d'un voyageur

La seconde couche est basée sur une approche réactive de recherche de chemin, de dé-

placement et d'évitement de collisions. Elle représente le niveau opérationnel de décision d'un piéton défini par [Daamen, 2004]. Si un voyageur cherche à atteindre une position donnée par la première couche du comportement, alors il s'y dirige tout en évitant les obstacles (*e.g.* les autres voyageurs, les murs, *etc.*).

Lorsqu'un voyageur se déplace, son comportement se base sur trois forces : (i) la force d'attraction vers l'objectif ; (ii) la force résultante des forces de répulsion des obstacles de l'environnement ; (iii) la force résultante des forces de répulsion issues des autres voyageurs.

Force $\overrightarrow{F}_{obj}$ qui tracte l'agent i vers son prochain objectif local :

$$\overrightarrow{F}_{obj} = \beta_{obj}.\overrightarrow{A_iG} \tag{6.1}$$

Force $\overrightarrow{F}_{rep_{ij}}$ de répulsion de l'agent j sur l'agent i :

$$\overrightarrow{F}_{rep_{ij}} = \beta_{ij}.\frac{m_i.m_j}{\|\overrightarrow{d_{ij}}\|^2}.\overrightarrow{d_{ij}} \tag{6.2}$$

Force $\overrightarrow{F}_{rep_{ik}}$ de répulsion de l'obstacle k sur l'agent i :

$$\overrightarrow{F}_{rep_{ik}} = \beta_{ik}.\frac{m_i}{(d_{ik}.\sin(\alpha_{ik}))^4}.\overrightarrow{n_{ik}} \tag{6.3}$$

avec
- A_i la position du piéton i.
- G la position du prochain objectif local du piéton i.
- O_k la position d'un obstacle k donné, situé dans l'espace de perçu du piéton i.
- $\overrightarrow{d_{ij}}$ le vecteur entre les piétons i et j, et $\|\overrightarrow{d_{ij}}\|$ la norme de ce vecteur.
- m_i masse du piéton i.
- $d_{ik} = \|\overrightarrow{A_iO_k}\|$, distance entre le piéton i et l'obstacle k.
- $\alpha_{ik} = \angle(\overrightarrow{A_iG}, \overrightarrow{A_iO_k})$
- $\overrightarrow{n_{ik}} = \begin{pmatrix} 0 & -sign(\alpha_{ik}) \\ sign(\alpha_{ik}) & 0 \end{pmatrix}.\frac{\overrightarrow{A_iO_k}}{\|\overrightarrow{A_iO_k}\|}$
- $\beta_{obj}, \beta_{ij}, \beta_{ik}$ constantes de calibrage.

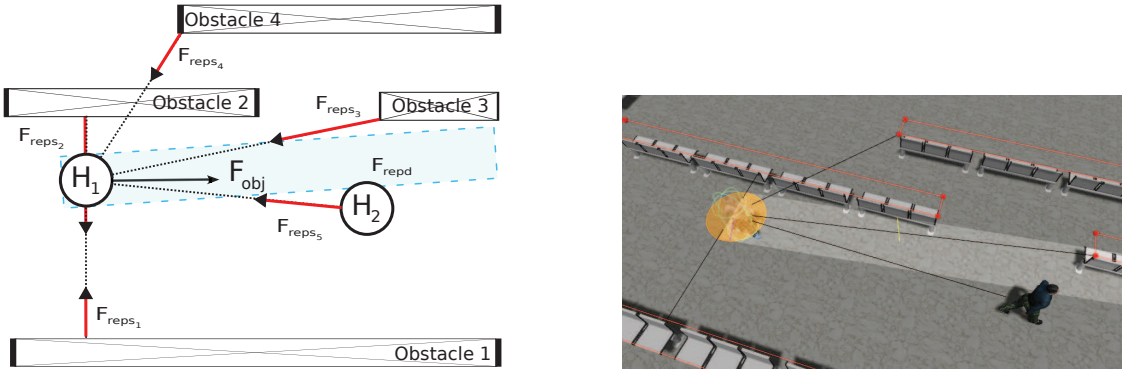
La figure 6.4 illustre les forces appliquées à un holon piéton H_1. Ici, le voyageur perçoit quatre groupes de bancs et un autre voyageur. Le point d'application de la force de répulsion d'un obstacle est le point le plus proche du holon H_i sur la surface de cet obstacle. Le déplacement de H_1 est le résultat de la somme des forces de répulsion des agents et des obstacles, et de la force de traction de l'objectif.

6.3.3/ MODÉLISATION DE L'ENVIRONNEMENT

Dans cette section, nous proposons un modèle concret de l'environnement pour la simulation du terminal de l'aéroport et décrivons les choix de conception les plus significatifs.

Rappelons que, dans ce manuscrit, nous ne traitons pas de la problématique de création du modèle graphique 3D d'un terminal d'aéroport. Nous utilisons les modèles 3D créés par notre partenaire industriel Voxelia SAS [2]. Ces modèles sont utilisés pour construire

2. http://www.voxelia.com



(a) Vue schématique

(b) Représentation des forces dans l'univers 3D

FIGURE 6.4 – Forces appliquées au holon H_1

l'ensemble des objets de l'environnement et pour initialiser leurs attributs (forme géométrique, taille, *etc.*).

Dans un premier temps, nous proposons une instanciation du métamodèle CRIO pour la simulation du terminal. Cette instanciation définit la structure et la décomposition statique de l'environnement en halls.

Les halls couvrant un espace important, nous appliquons une décomposition dynamique sur chacun d'eux. Cette décomposition est basée sur l'utilisation d'un arbre spatial et permet de limiter les temps de recherche des objets en fonction de critères spatiaux.

Nous discutons également de l'utilisation d'un modèle de sol permettant de déporter une partie des calculs des forces définies dans le comportement des voyageurs dans le modèle de l'environnement.

Nous poursuivons par une présentation de la holarchie de la simulation. Cette holarchie permet de simuler les comportements des voyageurs d'une part. D'autre part, elle exécute les missions dévolues à l'environnement. Cette holarchie est construite à partir des deux décompositions également présentées dans cette section.

Il s'ensuit une description des indicateurs dédiés à la simulation du terminal (*e.g.* l'énergie et les constantes de masse) définis au chapitre précédant (*cf.* section 5.3.2.1). Ces indicateurs sont utilisés par les holons pour déterminer la décomposition dynamique durant la simulation.

6.3.3.1/ DESCRIPTION DE LA STRUCTURE DE L'ENVIRONNEMENT

Le modèle concret de l'environnement du terminal de l'aéroport correspond à une instance du métamodèle CRIO proposée dans les chapitres précédents. La figure 6.5 présente une partie du diagramme UML d'objets décrivant cet instance du métamodèle.

Cette figure fait apparaître la décomposition statique en trois halls. Chaque zone est liée à un modèle de sol. La première utilité du sol est de permettre l'application de la gravité sur l'ensemble des objets de l'environnement. Dans la suite de ce chapitre, nous décrivons un usage alternatif permettant de diminuer les coûts de calcul dans les comportements des voyageurs. Chaque zone contient l'ensemble des objets localisés en son sein. Par exemple, les `Hall 1` et `Hall 2` contiennent les objets correspondant aux PIF. La

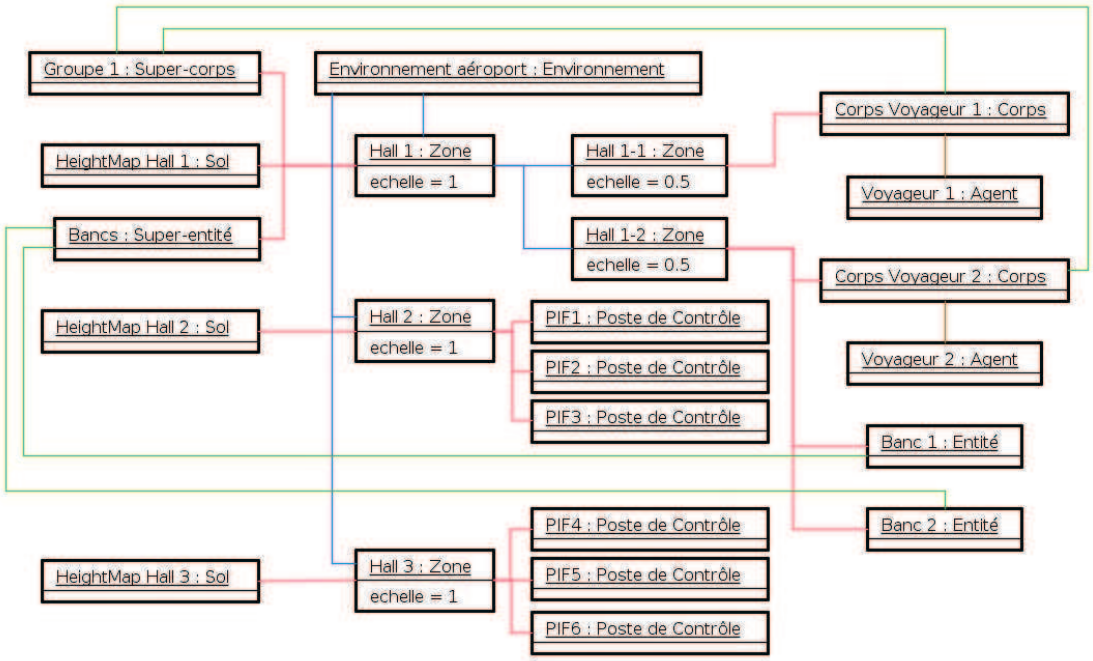


FIGURE 6.5 – Diagramme d'objets du métamodèle CRIO pour la simulation d'un terminal d'aéroport.

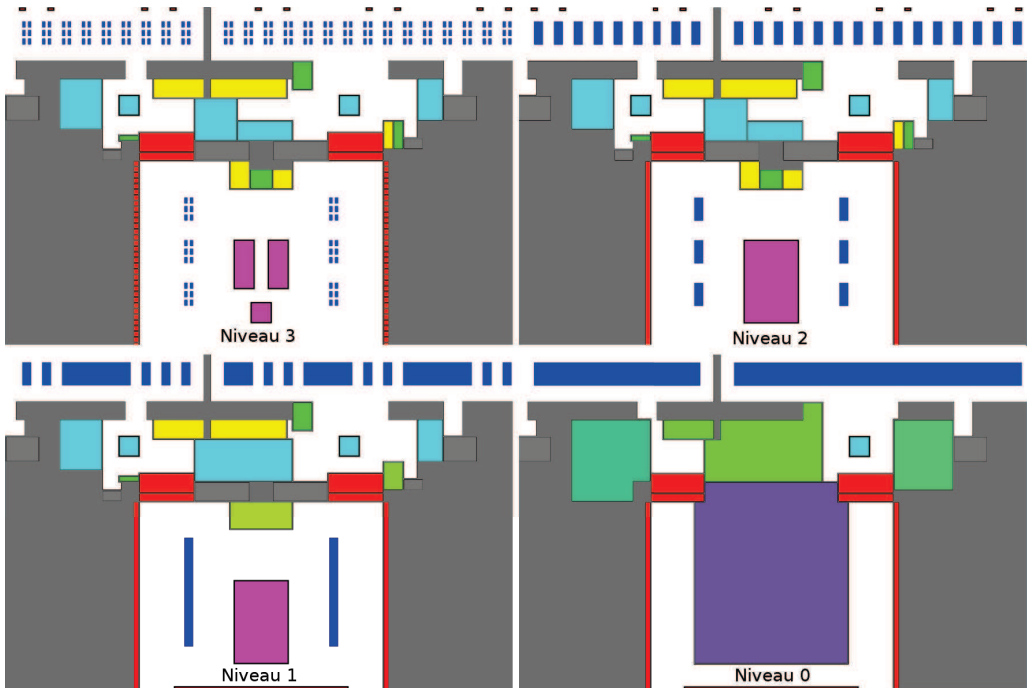


FIGURE 6.6 – Représentation de quatre niveaux de décomposition de l'environnement et des regroupements d'objets

figure 6.5 illustre également la décomposition dynamique du `Hall 1` en deux sous-zones.

Cette décomposition dynamique est basée sur l'utilisation d'un arbre spatial, décrit dans la sous-section suivante. Chacune des sous-zones contient les objets atomiques correspondant aux corps de voyageurs et au mobilier du terminal présents dans ces espaces. Une super-entité et un super-corps sont créés puis liés au `Hall 1`. Ils représentent un regroupement de bancs et un regroupement des corps de voyageurs, respectivement.

La figure 6.6 illustre différents regroupements d'objets définis dans notre modèle concret. Quatre niveaux de décomposition du terminal sont présentés sur la figure. Par exemple, les bancs se trouvant dans les halls d'embarquement, représentés par les petits rectangles bleus en haut des illustrations, sont regroupés par îlots, qui sont à leur tour regroupés dans le niveau supérieur de la hiérarchie de l'environnement. Le regroupement des objets, y compris des corps des voyageurs, est dépendant de la construction des niveaux de la hiérarchie de l'environnement et des indicateurs multiniveaux, tous deux présentés dans la suite de ce manuscrit. Rappelons qu'une super-entité est la représentation à un niveau d'abstraction n d'un ensemble d'objets du niveau d'abstraction $n+1$.

Dans la section suivante, nous décrivons le modèle de décomposition dynamique utilisé dans notre application.

6.3.3.2/ Description de la décomposition dynamique de l'environnement

Dans la section précédente, la décomposition statique de l'environnement est définie. Les espaces correspondant à chaque hall restent toutefois trop grand pour permettre une simulation performante. Il est nécessaire de décomposer ces espaces afin de converger plus rapidement lors de la recherche d'objets. Cette nouvelle décomposition est réalisée dynamiquement et dépends des décisions prises par les holons de l'environnement, décrits dans la suite de ce chapitre. Toutefois, nous pouvons préciser la méthode et les algorithmes qui sont utilisés pour réaliser cette décomposition dynamique.

La base de données des objets de l'environnement est l'un des éléments centraux dans le simulateur. Elle fournit les moyens pour localiser et manipuler chaque entité dans l'univers 3D. Cette base de données, appelée modèle du monde, est définie afin de fournir les meilleures performances possibles lors de la recherche des objets à partir de critères spatiaux. Il existe une instance de ce modèle de monde pour chacune des zones définies statiquement.

Lors d'un parcours dans un arbre spatial, les nœuds sont parcourus s'ils recouvrent une zone de l'espace correspondant à l'espace de recherche. Dans le cas contraire, ils sont écartés. Cette méthode permet d'écarter rapidement les zones (et leurs objets) pour lesquels il est sûr qu'ils n'appartiennent pas à l'espace de recherche. Dans les zones entrant en intersection avec l'espace de recherche, chaque objet rencontré dans un nœud doit faire l'objet d'un test d'intersection géométrique avec l'espace de recherche.

Un arbre spatial utilise une heuristique de partition pour hiérarchiquement décomposer l'espace. Le modèle JaSIM fournit différents modèles (et leurs implantations) d'arbres traditionnellement utilisés dans le domaine du jeu vidéo : arbre « *Binary Space Partition* » (BSP), *quadtree*, et *octree*. Ces arbres divisent respectivement l'espace en deux, quatre et huit sous-espaces.

Un axe de séparation (ou axe de partition) est un axe permettant de subdiviser l'espace en un ensemble d'espaces plus petits. Nous considérons qu'il est toujours parallèle à l'un des axes du repère géométrique global de l'univers 3D. Ce choix d'implantation est

introduit pour simplifier et améliorer les performances du processus de construction des arbres spatiaux [Galland et al., 2009].

Deux points essentiels ont une grande influence sur les performances des arbres spatiaux : (i) la position des axes de partition, et (ii) l'heuristique utilisée pour placer dans l'arbre les entités qui sont en intersection avec un ou plusieurs axes.

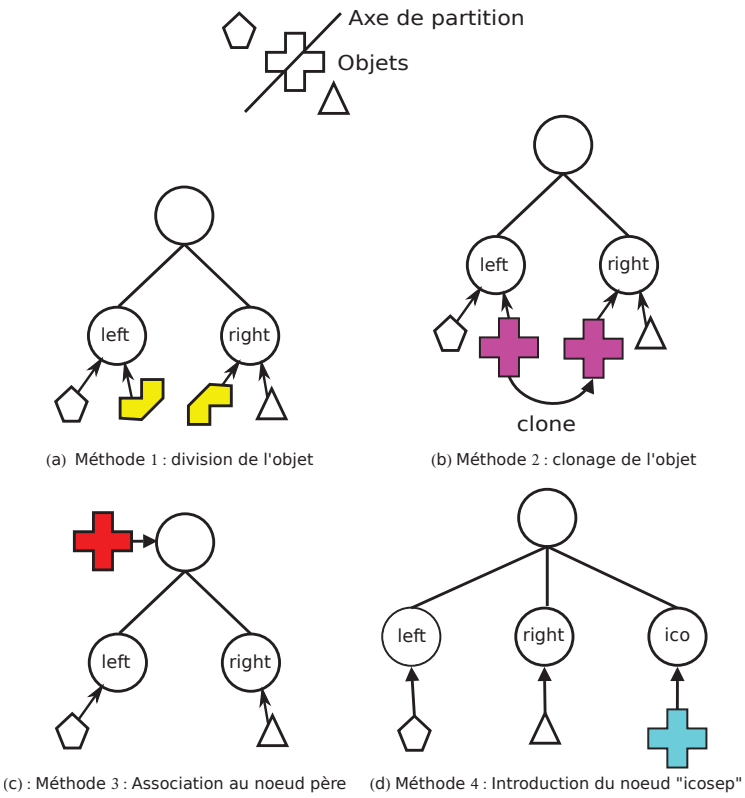
Généralement, les axes de partition sont placés au centre spatial de la zone ou au barycentre des objets situés dans cette zone. Nous choisissons la première approche pour sa simplicité d'implantation et son coût de calcul plus faible. En effet, l'ensemble des objets dans une zone évoluent continuellement. Il serait alors nécessaire de calculer le barycentre, puis de subdiviser à nouveau l'espace à chaque instant de la simulation. Ces deux procédures possèdent les complexités $O(n)$ et $O(n.\log n)$, respectivement. Il est alors évident qu'un axe de partition indépendant de l'ensemble des objets est plus efficace en termes de temps de calcul.

Quatre méthodes peuvent être utilisées pour placer dans l'arbre un objet en intersection avec un axe de partition. Les trois premières sont communément admises dans la littérature. La quatrième est utilisée par les modèles de détection de collisions entre objets 3D. Nous la proposons comme une alternative viable pour le modèle de l'environnement [Galland et al., 2009]. La figure 6.7 illustre ces quatre méthodes. Dans cet exemple, trois objets sont situés dans l'environnement. La partie supérieure de la figure représente leurs emplacements et l'axe de partition considéré. Les quatre sous-figures illustrent chacune des méthodes pour classer les objets en intersection avec l'axe de partition. Ces méthodes sont :

1. Le maillage 3D de l'objet est divisé en plusieurs parties par l'axe de partition. Chacune de ces parties est alors placée individuellement dans l'arbre.
2. L'objet est cloné. Chaque instance ainsi obtenue est placée dans la sous-arborescence.
3. L'objet n'étant pas entièrement englobé ni par la zone associée à la sous-arborescence de gauche, ni par la zone associée à celle de droite, il reste associé au nœud père.
4. L'objet est ajouté dans un nœud spécifique, appelé nœud « *icosep* » [Shagam, 2003].

Les méthodes 1 et 2 sont trop coûteuses en temps de calcul. D'une part, les opérations de division d'un maillage 3D et de fusion des parties nécessitent de traiter plusieurs centaines de primitives géométriques (triangles) pour chaque objet. D'autre part, retrouver l'ensemble des clones se trouvant dans un arbre nécessite de parcourir l'ensemble des nœuds constituant la zone englobant les clones. Cette sous-arborescence est alors utilisée comme une liste d'objets. La complexité de l'algorithme de recherche passe de $O(\log n)$ à $O(n)$ dans le pire des cas. De plus, l'utilisation de clones nécessite de filtrer l'ensemble des perceptions fournies aux piétons afin qu'un seul des clones apparaisse dans cet ensemble.

La méthode 4 est préférée à la méthode 3 car elle permet de réaliser moins de tests d'intersection. En effet, les nœuds « *icosep* » ne sont pas systématiquement parcourus et permettent d'écarter un plus grand nombre d'objets. [Shagam, 2003] propose d'ajouter un nœud « *icosep* » pour chaque cas d'intersection avec un ou plusieurs axes de partition, soit 19 nœuds « *icosep* » pour un arbre *octree* (divisant l'espace en 8 sous-espaces). Plus le nombre de nœuds « *icosep* » est grand, plus la probabilité de l'écarter durant le processus de recherche est importante.



(a) Méthode 1 : division de l'objet (b) Méthode 2 : clonage de l'objet

(c) : Méthode 3 : Association au noeud père (d) Méthode 4 : Introduction du noeud "icosep"

FIGURE 6.7 – Exemples des méthodes d'insertion d'objets dans un arbre BSP

L'annexe B présente les algorithmes utilisés par le simulateur pour définir et manipuler les arbres spatiaux.

Enfin, chaque nœud de l'arbre spatial est associé à un holon de la holarchie de l'environnement. À chaque fois qu'une zone doit être décomposée dans l'arbre spatial, le holon associé est lui aussi décomposé et des sous-holons sont créés. De même, si un holon décide de se décomposer, l'arbre spatial associé à la zone qu'il gère est lui aussi décomposé. Le processus de composition suit également ce principe.

6.3.3.3/ DESCRIPTION DU MODÈLE DE SOL

Dans cette sous-section, nous discutons l'utilisation d'un modèle de sol permettant de déporter une partie des calculs des forces depuis les comportements des voyageurs dans le modèle de l'environnement. Ce choix de conception améliore les performances globales du simulateur.

Dans la plateforme JASIM, le modèle de sol est une carte des hauteurs représentée par une matrice rectangulaire à deux dimensions. Il s'agit d'une représentation discrétisée du terrain : chaque cellule de la matrice contient la hauteur moyenne d'une petite partie du terrain. Pour des commodités de stockage et d'usage, une carte de hauteur est en général stockée dans une image "bitmap". Chaque pixel encode la valeur de la hauteur dans une de ses composantes de couleur.

Afin d'améliorer les performances du simulateur, nous adaptons le modèle du sol pour dé-

porter une partie des calculs des forces du comportement d'un voyageur dans le modèle de l'environnement. Nous considérons que les forces de répulsion des murs peuvent être calculées à priori et directement acquises/perçues et utilisées par les voyageurs. Pour cela, nous étendons le modèle de sol en y introduisant les forces de répulsion des murs.

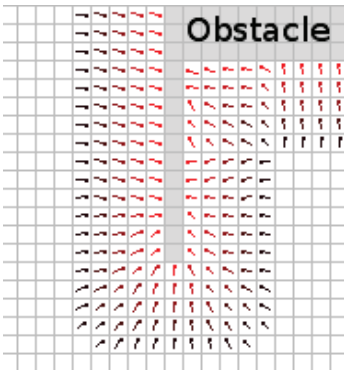


FIGURE 6.8 – Vue schématique des vecteurs de répulsion stockées dans une description du sol, avec $d_w = 5$

Ces vecteurs permettent aux holons piétons d'éviter les zones inaccessibles du terrain. La figure 6.8 illustre ce champs de répulsion. La zone grisée correspond à une zone inaccessible. Par exemple, une telle zone peut être utilisée pour empêcher les piétons de traverser les murs. Lorsque les holons piétons perçoivent le sol, ils ont accès à la description du champs de répulsion et peuvent l'intégrer dans leur algorithme de déplacement, et ainsi préserver des ressources de calcul. En effet, avec cette implantation, il n'est pas nécessaire d'ajouter les murs de l'aéroport dans l'ensemble des objets de l'environnement. Ces murs sont alors écartés des ensembles des objets perçus par les piétons. La force de répulsion associée à une cellule du sol aux coordonnées (i, j) est donnée par l'équation 6.4. i_o et j_o sont les coordonnées d'une cellule contenant un obstacle. Notons que nous considérons qu'un mur n'est plus « *répulsif* » s'il est éloigné d'un voyageur. Cette distance est représentée par d_w et corresponds au rayon d'influence d'un mur exprimé en nombre de cellules.

$$r_{ij} = \sum_{o \in \text{Obstacle}} \frac{(i,j) - (i_o, j_o)}{\left|(i,j) - (i_o, j_o)\right|} \times \frac{d_w}{|i-j| + |i_o - j_o|} \tag{6.4}$$

Dans la section suivante, nous présentons la holarchie de l'environnement. Cette holarchie permet de simuler les différentes missions de l'environnement du terminal d'aéroport et de mettre en œuvre la politique d'exécution multiniveau décrite dans les chapitres précédents.

6.3.3.4/ DESCRIPTION DE LA HOLARCHIE DE L'ENVIRONNEMENT

Cette section décrit le modèle concret de simulation multiniveau de l'environnement du terminal d'aéroport. Chaque niveau dans la décomposition de l'environnement est associé à un holon gérant la dynamique de la zone. Cette relation permet de créer une holarchie parallèlement à la décomposition hiérarchique de l'environnement présentée dans les sous-sections précédentes.

La figure 6.9 décrit la structure de l'environnement utilisée pour la simulation du terminal de l'aéroport. Cette structure est directement issue de la description de l'environnement dans la section 6.3.1. Le holon H_1 représente l'environnement du terminal. Les holons H_2 et H_3 sont des voyageurs. Dans l'application présentée, la simulation des voyageurs est faite au niveau microscopique. Les travaux de [Gaud, 2007] et [Razavi et al., 2011a] fournissent des modèles concrets de simulation multiniveau de piétons pouvant remplacer celui utilisé dans ce chapitre. Les holons H_4, H_5 et H_6 gèrent respectivement les trois halls du terminal. Durant le processus de simulation, ils se décomposent en accord avec les indicateurs énergétiques décrit dans la suite de ce chapitre.

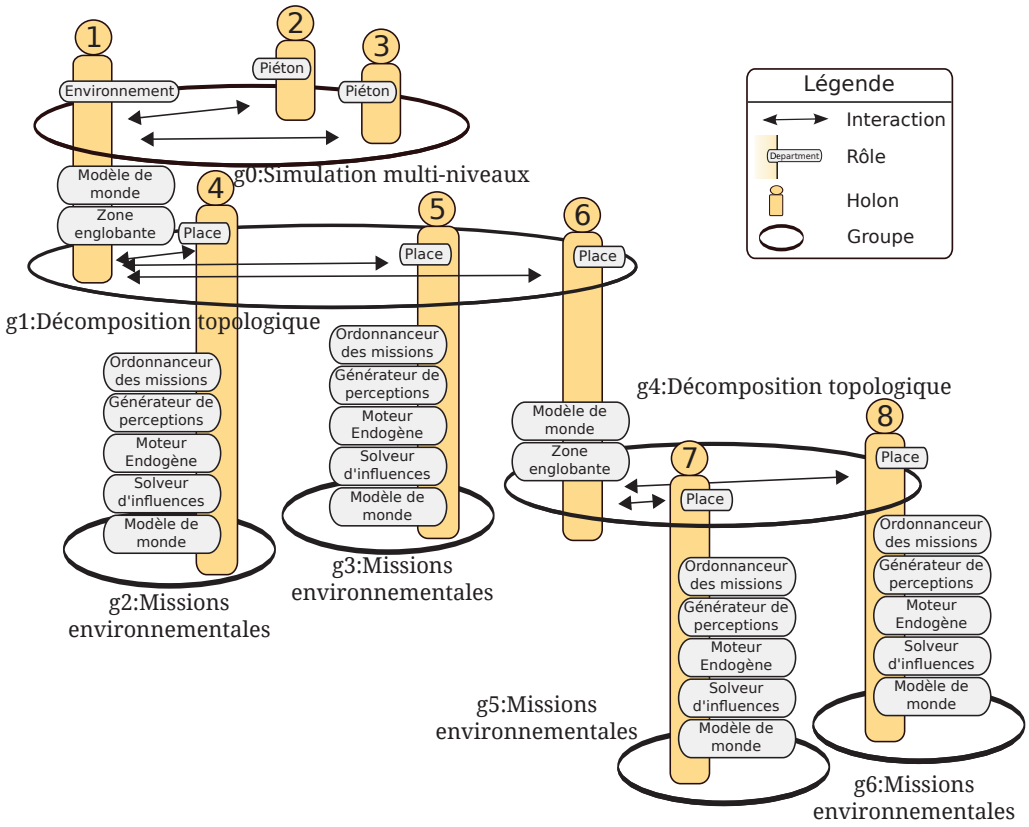


FIGURE 6.9 – Holarchie pour la simulation du terminal de l'aéroport

Ayant choisi de ne pas simuler les voyageurs par une holarchie, notre modèle de l'environnement doit fournir des perceptions microscopiques aux holons jouant le rôle `Piéton`. Notre modèle holonique de l'environnement est capable de créer des super-entités. Ces super-entités sont des regroupements d'objets de l'environnement et permettent un accès direct à toutes les entités atomiques le composant. Cette hypothèse d'implantation autorise la constitution d'un ensemble d'objets atomiques qui est alors fourni aux agents voyageurs. Ainsi, les voyageurs peuvent percevoir les objets atomiques en plus de leurs regroupements.

Les figures 6.10 et 6.11 représentent un exemple d'instanciation du modèle holonique pour notre application. Dans cet exemple d'instanciation, nous avons deux piétons au sein d'un environnement.

La figure 6.10 montre des instances des groupes *Simulation multiniveau* et *Décomposi-*

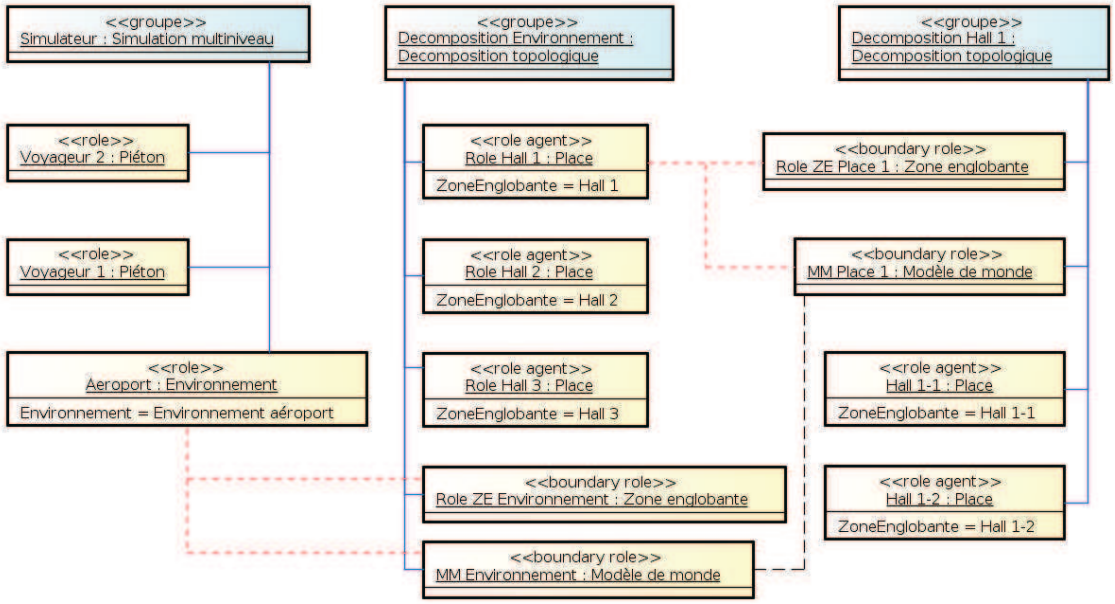


FIGURE 6.10 – Fragment du diagramme d'objets de la décomposition structurelle dans la holarchie de l'environnement

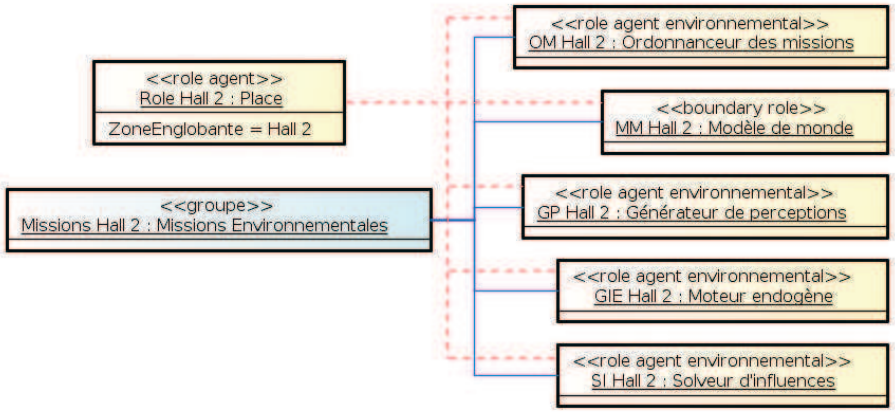


FIGURE 6.11 – Fragment du diagramme d'objets des missions de l'environnement dans la holarchie de l'environnement

tion topologique, ainsi que des rôles qui leurs sont associés. La figure 6.11 complète la figure précédente en détaillant les liens entre les instances des groupes et des rôles de la décomposition fonctionnelle (*i.e.* le groupe *missions environnementales* et ces rôles).

Dans ces figures, les liens bleus (continues) signifient que les instances des rôles sont associées à une instance d'un groupe. Les liens rouges (avec de petits pointillés) informent que les instances des rôles sont joués par la même instance d'un agent environnementale. Enfin, le lien noir (avec de grands pointillés) indiquent qu'un *modèle de monde* est un sous-ensemble d'un autre *modèle de monde* de niveau supérieur. Les instances des rôles `place` possèdent tous une instance du concept `zone englobante`, alors que l'instance du rôle `Environnement` possède une instance du concept *Environnement*, nommé

`Environnement aéroport`. Les instances des concepts ont déjà été détaillées dans la section 6.3.3.1, page 112.

Dans les deux sections suivantes, les indicateurs énergétiques utilisés durant la simulation par les holons de l'environnement sont présentés.

6.3.3.5/ INDICATEURS ÉNERGÉTIQUES

Cette section est dédiée à la description des indicateurs énergétiques aidant au changement de niveaux utilisés dans l'application de simulation du terminal de l'aéroport. Ils permettent d'évaluer la qualité de l'approximation fournie par un niveau d'abstraction plus élevé que le niveau microscopique.

Dans le chapitre 5, trois types d'énergie sont pris en compte : (i) l'énergie cinétique Ec_i, liée à la dynamique du holon i considéré ; (ii) l'énergie potentielle de l'objectif Epo_i qui est dépendante de l'objectif du holon i ; et (iii) l'énergie potentielle de contrainte Epc_i qui est liée aux éléments qui entravent la progression du holon i vers son objectif.

Sur la base de ces trois énergies, il est possible de calculer l'énergie globale d'un holon i dans la holarchie de l'environnement (*cf.* équation 5.1, page 101). Nous considérons alors qu'elle représente l'état courant de ce holon. Cette énergie permet de caractériser le comportement de chaque holon et d'évaluer l'approximation des comportements des sous-holons par le super-holon.

Nous considérerons dans ce chapitre que le calcul de ces énergies est dépendante de la qualité des perceptions à générer : plus les perceptions générées pour un holon sont différentes de celles attendues à un niveau inférieur (celles générées par ses sous-holons), plus l'énergie de ce holon augmente. Considérons l'ensemble des objets perçus e_H calculés par le super-holon et les ensembles similaires e_i calculés par les sous-holons. Les objets appartenant à ces ensembles peuvent être caractérisés ainsi :

- L'objet appartient à e_H et à au moins l'un des ensembles e_i. Il est nommé « *objet commun* » et appartient à l'ensemble noté $p^{\odot}$.
- L'objet appartient à e_H et à aucun des ensembles e_i. Il est nommé « *objet ajouté* » et appartient à l'ensemble noté $p^{\oplus}$.
- L'objet n'appartient pas à e_H et appartient à au moins un des ensembles e_i. Il est nommé « *objet perdu* » et appartient à l'ensemble noté $p^{\ominus}$.

L'objectif d'un holon de l'environnement est de fournir les meilleures perceptions possibles aux piétons. L'énergie associée est l'énergie potentielle de l'objectif Epo_i du holon i. Elle est définie par l'équation 6.5. Intuitivement, cette énergie évalue la qualité des perceptions générées : plus le nombre d'objets perdus est important par rapport au nombre d'objets communs, plus la qualité de la perception est faible.

$$Epo_i = \begin{cases} \dfrac{\alpha_{po}|p^{\ominus}| + \beta_{po}|p^{\oplus}|}{|p^{\odot}|} & \text{si } p^{\odot} \neq \emptyset \\ \alpha_{po}|p^{\ominus}| + \beta_{po}|p^{\oplus}| & \text{sinon} \end{cases} \tag{6.5}$$

α_{po} et β_{po} sont des variables de calibrages. Nous proposons les valeurs $\alpha_{po} = 1$ et $\beta_{po} = \frac{1}{|E|}$, où $|E|$ représente le nombre total d'entités dans le monde. Ce choix arbitraire nous permet de considérer que les objets perdus possèdent une plus grande influence sur l'énergie du super-holon que les objets ajoutés.

La dynamique d'un holon de l'environnement est représentée par l'évolution du champ de perception qu'il utilise pour calculer l'ensemble des objets perçus à son niveau d'abstraction. L'énergie associée est l'énergie cinétique Ec_i du holon i. Elle est définie par l'équation 6.6.

$$Ec_i = \begin{cases} \dfrac{\alpha_c|s^{\ominus}| + \beta_c|s^{\oplus}|}{|s^{\odot}|} & \text{si } s^{\odot} \neq \emptyset \\ \alpha_c|s^{\ominus}| + \beta_c|s^{\oplus}| & \text{sinon} \end{cases} \tag{6.6}$$

Les surfaces couvertes par les champs de perception sont caractérisées de manière similaire aux objets perçus dans la formule précédente : les *« surfaces ajoutées »* ($s^{\oplus}$), les *« surfaces supprimées »* ($s^{\ominus}$) et les *« surfaces conservées »* ($s^{\odot}$). Le calcul de ces surfaces est basé sur les formes géométriques des différents champs de perception projetées sur le plan du sol. α_c et β_c sont des variables de calibrages, auxquelles nous proposons les valeurs $\alpha_c = 1$ et $\beta_c = \frac{1}{S}$, où S représente la surface totale de l'environnement.

L'énergie potentielle de contrainte est liée aux filtres utilisés par la génération des perceptions. Ces filtres permettent d'écarter de l'ensemble des objets perçus les objets considérés comme inintéressants par le piéton. Ces filtres réduisent les perceptions générées qui seront fournis aux corps en fonction de critères donnés par les agents associés à ces corps. L'énergie associée est l'énergie potentielle de contrainte Epc_i du holon i. Elle est définie par l'équation 6.7.

$$Epc = \begin{cases} \dfrac{\alpha_{pc}.f^{\ominus} + \beta_{pc}.f^{\oplus}}{f^{\odot}} & \text{si } f^{\odot} \neq \emptyset \\ \alpha_{pc}.f^{\ominus} + \beta_{pc}.f^{\oplus} & \text{sinon} \end{cases} \tag{6.7}$$

Les filtres sont également caractérisés de manière similaire aux objets perçus dans la formule précédente : les *« filtres ajoutés »* ($f^{\oplus}$), les *« filtres supprimés »* ($f^{\ominus}$) et les *« filtres conservés »* ($f^{\odot}$). α_{pc} et β_{pc} sont elles aussi des variables de calibrages pour lesquelles nous proposons les valeurs $\alpha_{pc} = 1$ et $\beta_{pc} = \frac{1}{|F|}$, où $|F|$ représente le nombre de filtres différents.

6.3.3.6/ DÉFINITION DES AUTRES INDICATEURS MULTINIVEAUX

Dans l'approche proposée au chapitre 5, il est nécessaire de définir les constantes de masse pour tous les objets de l'environnement et pour toutes les zones. Dans notre application, nous considérons qu'il n'existe pas d'objet de l'environnement devant être priorisé dans le cadre de la simulation. De même, nous considérons que les trois zones de l'environnement (et leurs décompositions respectives) possèdent une importance égale dans la simulation. Quel que soit l'objet e et la zone z de l'environnement, nous définissons les constantes de masse ainsi : $w_e = 1$ et $w_z = 1$. Notre modèle définit également une ressource R conditionnant le choix du niveau de simulation. La constante k_z indique la quantité de ressource R consommée par chaque holon dans la holarchie environnementale pour exécuter ses algorithmes de sélection de niveau. Dans notre cas d'application, nous avons mesuré cette consommation telle que $k_z \leq 3ms$.

La valeur de R définit la ressource disponible pour réaliser la simulation. Dans la suite de ce chapitre, cette ressource représente le nombre de calculs de perception pouvant être réalisé pour un pas de la simulation et pour l'ensemble des entités simulées : $R = |E|$, avec

$|E|$ le nombre de voyageurs simulés à un instant donné. Ce choix arbitraire nous permet de nous abstraire d'indicateurs bas-niveaux plus difficiles à maîtriser dans la cadre de tests de performances : temps de calcul maximale, taux d'occupation des processeurs, *etc*.

À ce point du manuscrit, nous avons complètement défini le scénario de simulation du terminal de l'aéroport et les deux modèles qui sont utilisés : le modèle de comportement des voyageurs et le modèle de l'environnement. La section suivante présente succinctement l'implantation et le déploiement de ces modèles sur les plateformes JASIM et JANUS.

6.4/ IMPLANTATION ET DÉPLOIEMENT

Dans cette section, nous décrivons brièvement l'architecture logicielle utilisée et nos contributions à cette architecture. Enfin, nous proposons une solution de déploiement sur les plateformes JASIM et JANUS.

6.4.1/ EXTENSION DE LA PLATEFORME JASIM

Le modèle JASIM, et son implantation, est un projet du laboratoire IRTES-SET visant à fournir les modèles et les outils logiciels pour modéliser et simuler les individus dans des villes virtuelles. L'objectif de ce projet est de fournir la possibilité de simuler des piétons dans des environnements en trois dimensions. Ces environnements sont obtenus grâce à des systèmes d'informations géographiques 3D. Le modèle JASIM propose une description de la structure de l'environnement, de sa dynamique, ainsi que des modèles de comportement pour chaque individu peuplant la ville virtuelle (en relation avec les cas d'application étudiés). Une première version du modèle d'environnement JASIM et de son architecture a été proposée dans [Galland et al., 2009]. La particularité de ce modèle est qu'il ne propose aucun modèle organisationnel ou orientée-agent parmi les composants du modèle de l'environnement. Ainsi, notre contribution majeure à la plateforme JASIM est l'intégration de notre modèle organisationnel et holonique de l'environnement. Pour réaliser notre modèle, nous intégrons notre modèle organisationnel dans JASIM, car ce dernier est un modèle microscopique non organisationnel. Pour cela, nous associons à l'environnement et à chaque place un holon. La hiérarchie des holons est similaire à celle de la décomposition spatiale.

La figure 6.12 décrit l'architecture globale de la plateforme JASIM et les relations entre ses différents modules. La plateforme JASIM utilise la plateforme multiagent JANUS pour mettre en œuvre et exécuter les modèles d'agents. En effet, JANUS est une plateforme qui propose une implémentation du métamodèle CRIO et un support de la dynamique au sein de holons. Le module *« Object-oriented Environment Model »* propose un ensemble de modèles, de structures logicielles (orientées-objet) et d'algorithmes permettant d'exécuter les différentes missions de l'environnement. Les principaux algorithmes sont décrits dans l'annexe B, page 175. Un agent applicatif (dans notre cas, un voyageur) est une implantation d'un `Situated Agent` et interagit avec le modèle d'environnement de JASIM via son corps (*« 3D Body »*).

Notre modèle d'environnement holonique est intégré comme composant du module *« JaSIM-Janus Wrapper »* . Les concepts décrivant l'environnement (zone, bâtiment,

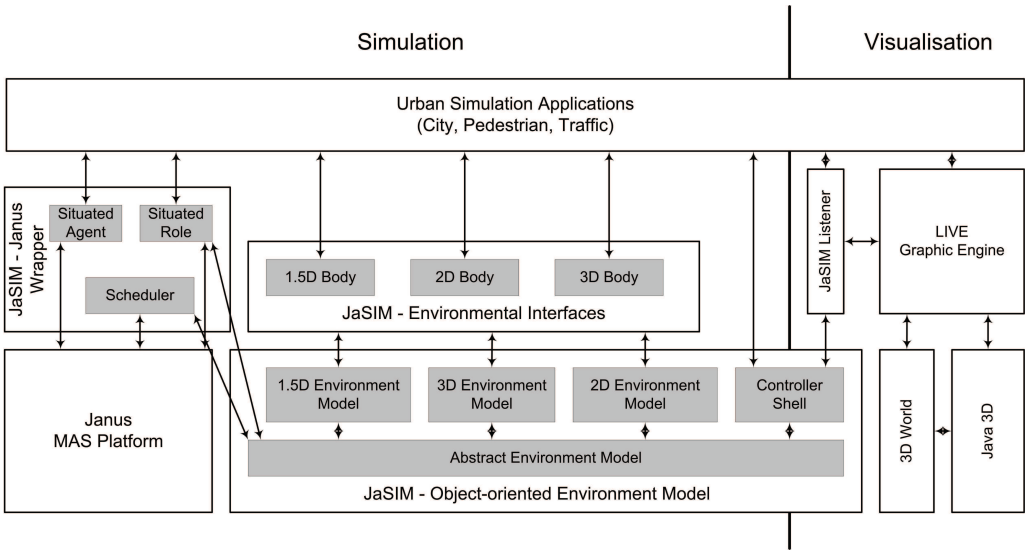


FIGURE 6.12 – Architecture globale de la plateforme JASIM permettant d'exécuter un SMA en relation avec un environnement virtuel [Galland et al., 2009]

étage, *etc.*), ainsi que les super-entités et les super-corps, ont été ajoutés dans la structure décrivant l'environnement. Cette structure est définie dans le module « *Object-oriented Environment Model* » .

6.4.2/ DÉPLOIEMENT DU MODÈLE

La figure 6.13 montre un exemple de déploiement de notre modèle d'environnement sur une machine virtuelle JANUS, dans le cadre de notre application de simulation d'un terminal d'aéroport.

Dans la plateforme JANUS, il est possible de choisir le mode d'exécution de chaque agent. Un mode d'exécution donne la relation entre le code exécutable d'un agent et les moyens utilisés pour exécuter ce code. Deux modes d'exécution sont fournis : un mode asynchrone et un mode synchrone. Le mode asynchrone se rapproche d'une exécution du code de l'agent en parallèle des autres agents en utilisant un processus lourd ou un processus léger (*thread*). Le mode d'exécution synchrone utilise un unique *thread*, appartenant à la machine virtuelle JANUS, pour exécuter séquentiellement le code des agents.

Les holons jouant le rôle `Place`, et associés aux zones issues de la décomposition statique de l'environnement, sont exécutés selon un mode asynchrone. Ainsi, nous avons la possibilité de paralléliser l'exécution d'une partie de notre modèle. Les autres holons, c'est-à-dire ceux jouant le rôle `Place` et n'étant pas associés avec une zone issue de la décomposition statique, sont exécutés avec la méthode asynchrone en priorité, s'il reste des *threads* disponibles dans la machine virtuelle JANUS. Dans le cas contraire, ils sont ajoutés dans la liste des holons à ordonnancer au sein du *thread* de la machine virtuelle JANUS.

Dans cette section, nous avons donné quelques détails sur les choix d'implantation et de déploiement du modèle de simulation. La section suivante permet d'évaluer la pertinence

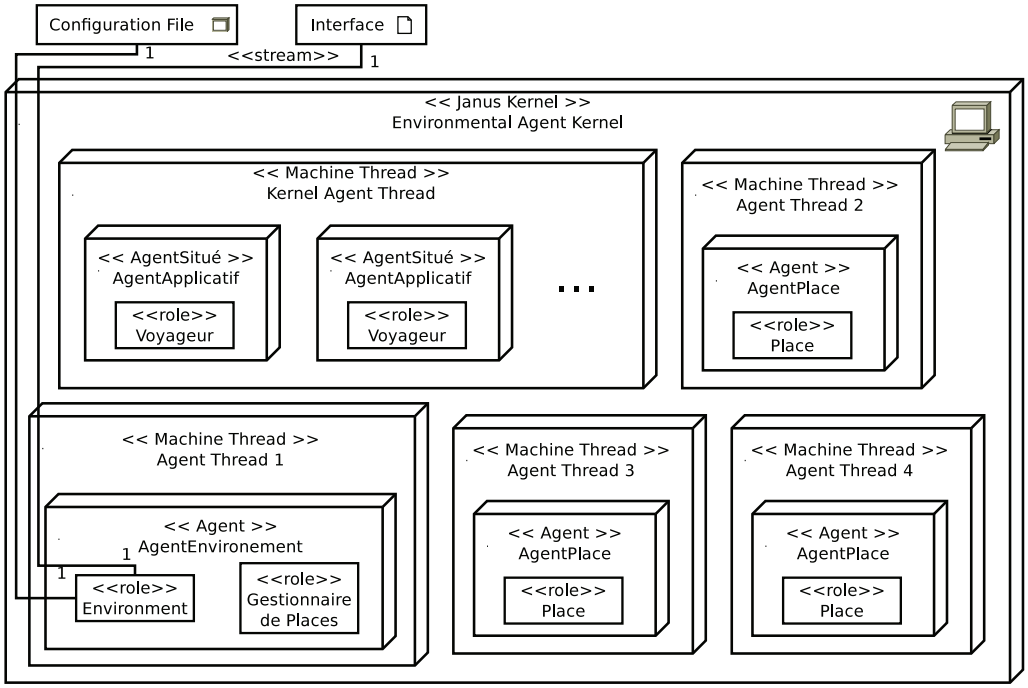


FIGURE 6.13 – Exemple de déploiement du modèle de simulation d'un terminal d'aéroport

de notre modèle par un ensemble d'expérimentations.

6.5/ RÉSULTATS EXPÉRIMENTAUX

Nous terminons ce chapitre par une section dédiée aux expérimentations suivant l'instanciation du modèle proposé dans cette thèse. Elles portent sur la simulation des halls d'un terminal d'aéroport et des PIF.

La plupart des tests ont été effectués sur un Intel® Core™i7 CPU Q740, 1,73 GHz avec 4 Go de mémoire RAM. Deux types d'évaluations sont présentées dans la suite de cette section : (i) l'évaluation des files d'attentes aux PIF, et (ii) l'évaluation des temps de calcul de l'implantation de notre modèle de simulation sur les plateformes JASIM et JANUS.

6.5.1/ ÉVALUATION DES FILES D'ATTENTE AUX POINTS DE CONTRÔLE

D'après [Jackou, 2010], *« les endroits bondés comme les files d'attente à l'enregistrement, le contrôle de sécurité et d'embarquement qui présentent une forte densité de personnes sont des cibles potentielles »* . Ainsi, nous analysons le nombre de voyageurs présents dans les files d'attente avant les PIF. La figure 6.14 présente l'évolution au cours du temps du nombre de voyageurs en attente pour aller aux halls d'embarquement.

Dans cette expérimentation, la capacité maximale des files d'attente est fixée à une centaine d'entités. Si le nombre de voyageurs est supérieur à cette borne, alors il n'y a pas suffisamment de PIF ouverts pour consommer les flux de passagers.

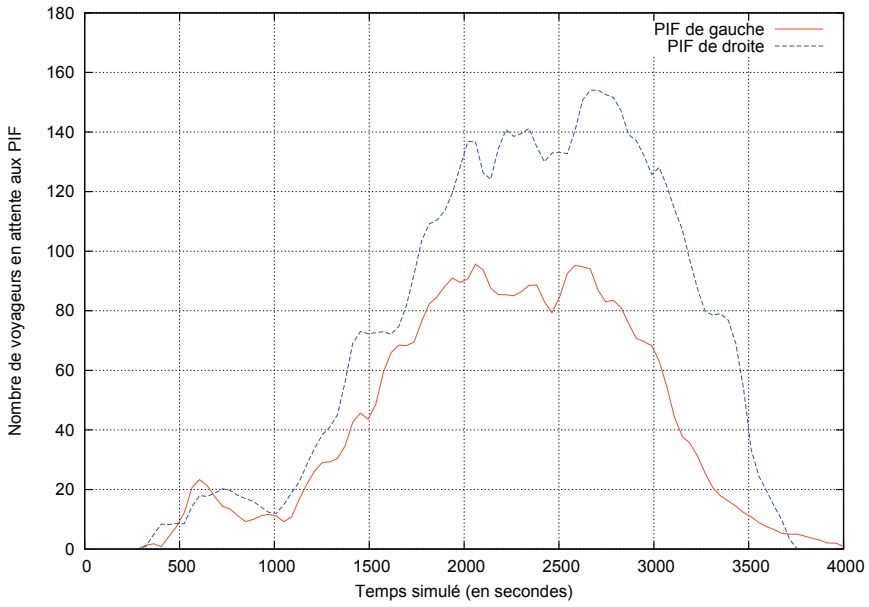


FIGURE 6.14 – Nombres de voyageurs dans les files d'attente avant les PIF.

Les passagers en attente aux PIF pour aller au hall d'embarquement de droite sont moins nombreux que ceux attendant aux PIF permettant d'aller au hall d'embarquement de gauche. Le nombre de voyageurs dans la file d'attente, pour les PIF de gauche, ne dépasse pas la centaine. Au contraire, pour la file des PIF de droite, ce nombre dépasse la centaine pendant environ 25 minutes. Dans ce cas, deux choix sont possibles, soit effectuer moins de contrôles, soit ouvrir des PIF supplémentaires. Étant donné l'importance de la sécurité au sein des aéroports, le premier choix est souvent privilégié [McLay et al., 2009].

La figure 6.15 illustre les temps moyens que met un passager pour passer du hall d'enregistrement au hall d'embarquement.

Le temps moyen d'attente des passagers est impacté par la taille de la file d'attente. Toutefois, comme nous pouvons le constater sur la courbe rouge, ce temps ne peut pas être calculé directement en fonction de la taille de la file d'attente. Ici, l'attente de quelques passagers reste importante malgré le faible nombre de voyageurs dans la file d'attente. Cette attente est dû à un phénomène de congestion causé par la lenteur des contrôles effectuées sur les derniers voyageurs.

La suite des expérimentations se concentre sur l'évaluation de notre modèle.

6.5.2/ ÉVALUATION DES TEMPS DE CALCUL DES MISSIONS ENVIRONNEMENTALES COÛTEUSES

Les deux missions de l'environnement que sont la génération des dynamiques endogènes et l'application des actions sont considérées comme négligeables en temps de

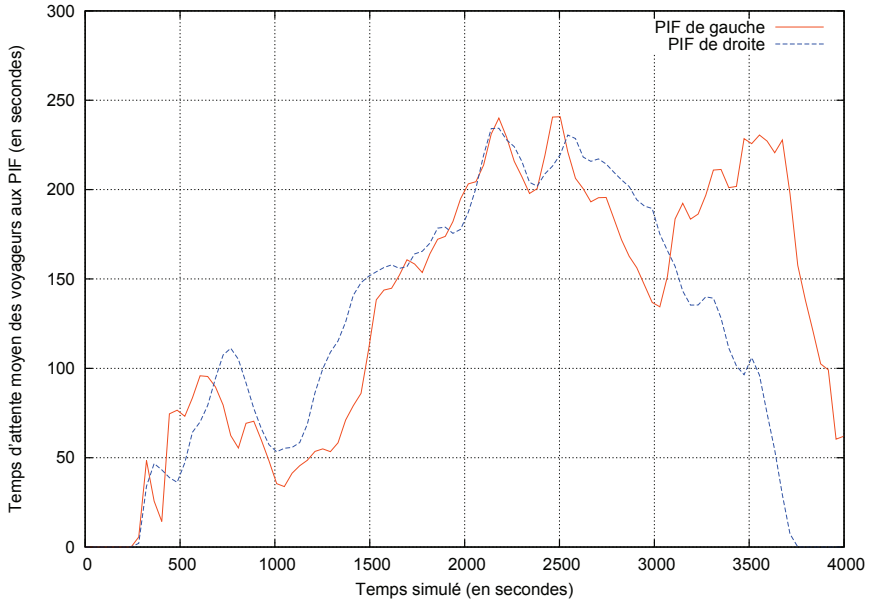


FIGURE 6.15 – Temps moyen d'attente des passagers aux PIF.

calcul dans notre application [Galland et al., 2009]. L'évaluation des temps de calcul de la simulation se focalise donc sur les missions de génération des perceptions et de mise à jour du modèle de monde. L'évaluation présentée dans cette section a pour objectif l'étude des performances de notre modèle et des coûts de calcul associés. Plusieurs scénarios d'évaluation sont considérés : (i) l'évaluation suivant la décomposition de l'environnement en une unique zone ; (ii) l'évaluation suivant la décomposition de l'environnement en trois zones ; et (iii) l'évaluation du générateur de perceptions suivant les ressources accordées pour la simulation.

Nous étudions particulièrement le temps nécessaire à l'exécution de la mission environnementale de génération des perceptions. En effet, cette mission est considérée comme la plus coûteuse en terme de ressources de calcul lors de l'exécution du modèle de l'environnement [Galland et al., 2009].

Les temps de calculs sont données en millisecondes. Les résultats des scénarios d'évaluations sont respectivement présentés dans les sections qui suivent.

6.5.2.1/ SIMULATION MICROSCOPIQUE D'UN ENVIRONNEMENT COMPOSÉ D'UNE ZONE

Dans ce scénario d'évaluation, l'application est adaptée pour utiliser une décomposition de l'environnement en une seule zone. Les perceptions pour un ensemble de 2000 entités sont générées en $865,83$ millisecondes avec un écart-type de $263,46$ millisecondes. La mise à jour des structures de données de l'environnement est beaucoup moins coûteuse : $16,59$ millisecondes avec un écart-type de $6,21$ millisecondes. L'initialisation et la montée en charge du modèle ne sont pas prises en compte dans les évaluations précé-

dentes. Ainsi, les 2000 entités initialement présentes dans l'environnement sont créées en $1640,45$ millisecondes.

Ces évaluations ont été obtenues lors de la simulation du cas d'étude présenté dans ce chapitre. Quelques facteurs expliquent le coût relativement élevé de la génération des perceptions. Le premier facteur est le choix de la méthode de décomposition dynamique de l'environnement, présenté dans la section 6.3.3.2 page 114. En effet, la complexité du calcul des perceptions pour l'arbre spatial choisi est de l'ordre de $n \log_5(n)$. En choisissant un arbre *« icosep »* [Shagam, 2003], la complexité moyenne du calcul des perceptions devient $n \log_{27}(n)$. Un arbre *« icosep »* est un octree (un arbre divisant récursivement l'espace en huit zones de même dimensions). Les objets entrant en intersection avec un ou plusieurs des axes de séparation d'un nœud est placé dans le nœud fils associé. L'ensemble des combinaisons de découpe de l'espace d'un arbre *« icosep »* fournit un ensemble de 19 régions différentes. Un facteur supplémentaire influençant légèrement les résultats est la génération des nombreux fichiers contenant les données d'acquisitions.

6.5.2.2/ SIMULATION MICROSCOPIQUE D'UN ENVIRONNEMENT COMPOSÉ DE TROIS ZONES

Dans ce scénario d'évaluation, l'environnement est composé de trois zones. Les tables [3] 6.1 et 6.2 montrent les résultats des temps de calcul de la génération des perceptions et de la mise à jour des structures de données, respectivement .

	Hall d'enregistrement avec 1000 passagers (a)	Hall d'embarquement à droite avec 600 passagers (b)	Hall d'embarquement à gauche avec 400 passagers (c)	$a+b+c$	$\max(a,b,c)$	Environnement
Temps Moyen	$586,23$	$406,96$	$283,29$	$1276,48$	$590,89$	$600,47$
Écart Type	$201,92$	$162,74$	$121,77$	$432,15$	$203,52$	$209,15$

TABLE 6.1 – Temps moyens de calcul (en millisecondes) pour la mission "*génération des perceptions*" au niveau microscopique.

Le hall d'enregistrement contient en moyenne plus de passagers que les autres halls. Par conséquent, les missions associées à cette zone prennent plus de temps. Les tables 6.1 et 6.2 illustrent également deux politiques d'exécutions des holons qui sont dépendantes de la machine. La colonne $a+b+c$ est le temps moyen de l'exécution de l'environnement lorsqu'une politique d'exécution synchrone est utilisée. La colonne $\max(a,b,c)$ est le temps moyen lorsqu'une politique d'exécution asynchrone est utilisée. La colonne "Environnement" présente le temps de calcul pour un pas de simulation pour l'ensemble du modèle de l'environnement. Notons que les valeurs affichées sont supérieures aux valeurs de la colonne $\max(a,b,c)$. Ce délai supplémentaire corresponds aux interactions entre les différents holons de l'environnement leurs permettant d'échanger des corps de passagers (lorsqu'un passager passe d'un hall à l'autre). Notons de cette différence,

3. Ces tables montrent uniquement les résultats de simulation pour l'environnement composé de trois zones

	Hall d'enregistrement avec 1000 passagers (a)	Hall d'embarquement droite avec 600 passagers (b)	Hall d'embarquement gauche avec 400 passagers (c)	$a+b+c$	$\max(a,b,c)$	Environnement
Temps Moyen	$10,21$	$6,53$	$4,29$	$21,03$	$10,54$	$15,20$
Écart Type	$4,32$	$3,37$	$2,49$	$7,77$	$4,34$	$5,55$

TABLE 6.2 – Temps moyens de calcul (en millisecondes) pour la mission "*mise-à-jour des structures topologiques*" au niveau microscopique.

qu'elle peut être utilisée pour initialiser la constante k_z dans notre modèle holonique de l'environnement.

Si nous comparons les résultats de cette expérimentation avec ceux présentés dans la section précédente, nous voyons apparaître un gain en terme de temps de calcul en faveur d'un modèle utilisant une décomposition de l'environnement en trois zones. Toutefois, ce gain est valable uniquement pour une exécution asynchrone.

6.5.2.3/ TEMPS DE CALCULS ET ÉNERGIE DES HOLONS POUR LA GÉNÉRATION DES PERCEPTIONS SUIVANT QUATRE RESSOURCES DIFFÉRENTES

Dans ce scénario d'évaluation, seule la mission de génération des perceptions est présentée. L'environnement est composé d'une unique zone. La machine utilisée pour ces tests possède un processeur Intel® Xeon® CPU E5430, 2,66 GHz avec 8 Go RAM. Cette machine nous a permis de supporter l'ensemble de nos calculs, et ce, pour un plus grand nombre d'entités. Notons également que ces tests sont effectués indépendamment de l'application.

Lors de la génération des perceptions, nous allouons une ressource R, telle que définie dans notre modèle holonique multiniveau, à la racine de la holarchie. Pour des raisons de simplification, la ressource R est assimilée au nombre maximal de perceptions à générer pour un pas de temps de la simulation. Nous considérerons différents cas où la ressource R est respectivement égale à 10%, 60%, 80% et 100% du nombre de passagers simulés. Ce dernier cas (100%) est assimilé à la simulation microscopique de l'ensemble du système sur un calculateur possédant suffisamment de ressources. Les trois premiers cas permettent de forcer le simulateur à mettre en œuvre la politique d'évaluation et d'exécution multiniveau.

Les passagers sont répartis uniformément dans un environnement vide. La densité ayant une influence sur les performances du simulateur, deux densités de population différentes sont utilisées durant les évaluations : 1/5 et 1/10. Autrement dit, 1/5 ou 1/10 de l'espace est occupé par la population. Les tables 6.3 et 6.4 présentent les temps moyens de calcul pour ces deux densités de population. N et R sont respectivement le nombre de passagers et le pourcentage de perceptions générées durant les tests.

Les performances affichées ne sont pas linéaires. Cette tendance s'explique par l'augmentation du temps de calcul pour la génération d'une perception d'un super-corps par rapport à celle sur un corps. Cette augmentation n'est pas linéaire, car une duplication

Density = 1/5				
N \ R	10%	60%	80%	100%
1000	2,931	10,628	13,725	17,006
2000	5,476	24,007	31,029	34,790
4000	14,507	56,198	70,577	82,510
6000	22,552	88,759	112,111	132,834
8000	32,912	137,933	178,248	210,741
10000	39,723	161,319	209,262	246,246
20000	100,602	454,443	588,980	700,228
40000	225,510	1079,970	1368,895	1553,022
60000	399,072	1835,486	2294,053	2887,258
80000	588,365	2634,125	3208,900	3688,020
100000	788,279	3557,496	4705,323	5374,101
200000	1906,037	8491,271	11159,395	12538,545
400000	4457,565	18626,260	24129,017	27632,890
600000	6861,324	29625,365	38447,508	45524,113
800000	10015,665	41253,278	53461,905	60608,785
1000000	13066,324	56514,569	75215,186	91605,405

TABLE 6.3 – Temps moyens de calcul (en millisecondes) avec une densité de population de 1/5.

Density = 1/10				
N \ R	10%	60%	80%	100%
1000	3,037	8,775	10,878	12,334
2000	4,946	18,923	23,331	25,841
4000	12,913	44,250	53,464	60,514
6000	21,717	71,279	87,075	96,671
8000	27,757	99,959	124,299	136,265
10000	37,195	128,594	155,155	174,717
20000	93,752	338,459	425,912	488,856
40000	199,925	785,205	986,403	1061,891
60000	349,170	1350,404	1628,111	1900,737
80000	508,925	1871,634	2284,346	2734,008
100000	686,297	2741,963	3247,932	3760,069
200000	1639,557	6305,206	8023,662	8854,911
400000	3864,251	14393,312	21200,919	21029,041
600000	5835,449	21632,444	27235,326	30692,983
800000	8571,180	30616,640	38443,476	43773,283
1000000	11931,204	42209,866	54715,589	59343,348

TABLE 6.4 – Temps moyens de calcul (en millisecondes) avec une densité de population de 1/10

des perceptions est ajoutée pour fournir à chaque voyageur une perception microscopique.

La simulation avec de faibles ressources implique une augmentation de l'approximation des résultats de la simulation. Plus exactement, cette augmentation est due à l'augmentation de l'énergie des holons (*cf.* section 6.3.3.5) exécutant les missions environnementales.

L'énergie des holons exécutant la génération des perceptions est dépendante de l'approximation faite sur un groupe de corps de piétons. Nous définissons l'approximation

d'un groupe de corps de piétons tel que : (i) le champs de perception du groupe correspond à la plus petite boite englobante recouvrant l'ensemble des champs de perception des corps des piétons constituant le groupe et ; (ii) la taille et la position du groupe sont équivalente à celles du champs de perception. Notons également que nous n'utilisons aucun filtre pour la génération des perceptions. Ainsi, l'énergie Epc est nulle. Par la suite, nous analysons uniquement l'énergie Epo, car l'énergie Ec reste négligeable par rapport à cette dernière. La cause provient du champs de perception d'un groupe de piétons qui recouvre l'ensemble des champs de perception de chaque piéton. L'énergie Epo est calculée suivant les perceptions générées entre un holons et ses sous-holons.

Rappelons que les perceptions générées pour le corps d'un groupe sont mésocopiques. Autrement dit, les groupes d'entités peuvent être perçus à la place des entités qui la composent. Ces groupes sont considérés comme des entités perceptibles à un niveau d'abstraction supérieur. Bien que les perceptions soient transformées en perceptions microscopiques pour les piétons, l'énergie est calculée en fonction de ces perceptions mésoscopiques.

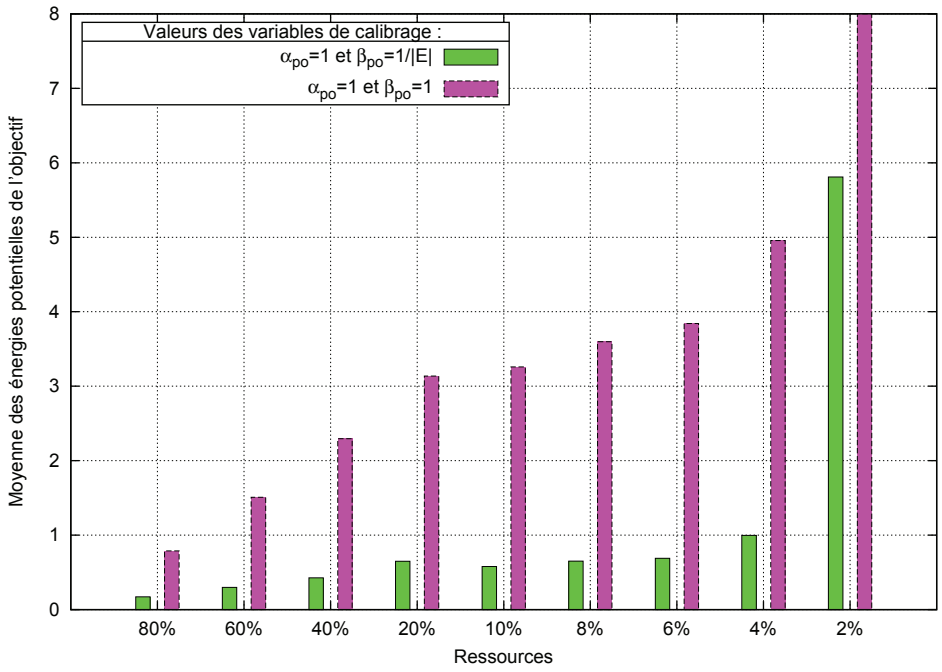


FIGURE 6.16 – Moyenne des énergies potentielles de l'objectif des holons exécutant la génération des perceptions.

La figure 6.16 illustre la moyenne des énergies potentielles de l'objectif des holons exécutant la génération des perceptions. Cette moyenne permet d'estimer si la simulation reste viable dans son ensemble.

Pour ces énergies, nous considérons une densité de population de 1/5. De plus, nous proposons deux jeux de tests suivant les variables de calibrages α_{po} et β_{po}. Pour une meilleure évaluation de l'énergie, nous testons sur les ressources R suivantes : 2%, 4%,

6%, 8%, 10%, 20%, 40%, 60% et 80% du nombre de passagers simulés.

D'après la figure 6.16, nous constatons que les variables de calibrage α_{po} et β_{po} n'ont pas d'influence sur la tendance de la courbe. En effet, cela provient de la corrélation entre les variables suivantes : $p^{\ominus}$ et $p^{\oplus}$. L'énergie augmente linéairement entre 100% et 20% puis se stabilise entre 20% et 6% et enfin augmente de manière exponentielle vers 4%. Les valeurs de l'énergie pour $\beta_{po} = 1$ sont plus grandes (environ quatre fois plus) que celles pour $\beta_{po} = 1/|E|$. Cela s'explique par une plus grande proportion d'objets ajoutés que d'objets perdus (*cf.* la définition des objets ajoutés et perdus, page 120). Les courbes montrent qu'en dessous d'un seuil (situé vers 4%), l'énergie d'un holon augmente exponentiellement. En considération de ces résultats, nous constatons que le rapport entre la précision et le temps calculatoire nécessaire à la génération des perceptions n'est plus intéressante en dessous de 4% du nombre de passagers simulés.

6.6/ CONCLUSION

Ce chapitre décrit la simulation d'un terminal aéroport pour l'évaluation des files d'attentes aux Postes d'Inspection Filtrage (PIF). Le modèle de simulation du terminal est composé de deux parties : le modèle de comportement des voyageurs, et le modèle de l'environnement.

Les voyageurs sont définis par un comportement semi-réactif composé de deux niveaux. Le premier niveau permet au voyageur de déterminer le prochain but à atteindre dans la séquence d'actions : s'enregistrer, passer à un point de contrôle et embarquer. Ces trois étapes constituent les principales phases du processus d'embarquement d'un voyageur. Le second niveau est basé sur une approche réactive de recherche de chemin, de déplacement et d'évitement d'obstacles. Si un voyageur cherche à atteindre une position dans l'aéroport donnée par le premier niveau de comportement, alors il s'y dirige tout en évitant les obstacles (*e.g.* les autres voyageurs, les murs, *etc.*).

Dans ce chapitre, nous décrivons l'instanciation du modèle de l'environnement proposé dans les chapitres précédents sur l'application de la simulation d'un terminal. La structure holarchique de l'environnement est discutée, et l'ensemble des indicateurs multiniveaux sont raffinés et définis pour la simulation du terminal.

Ce chapitre se termine par un ensemble d'évaluations réalisées sur le simulateur du terminal de l'aéroport. La première évaluation concernent l'étude des files d'attentes aux PIF. Nous présentons une évolution quantitative des tailles des files d'attentes, ainsi que des temps moyens d'attente des passagers pour passer les contrôles.

Une série d'évaluations des performances du simulateur est présentée en fin de chapitre. Ces évaluations permettent d'estimer les temps de calcul du simulateur que nous proposons dans cette thèse. Les évaluations sont réalisées avec et sans décomposition de l'environnement en zones, et également avec et sans l'existence de plusieurs niveaux d'abstraction dans l'environnement. De plus, une évaluation de l'approximation des holons est donnée par le biais de la valeur moyenne des énergies des holons exécutant les missions environnementales. Cette évaluation tend à confirmer l'utilisation de notre approximation jusqu'à un certain seuil.

La conclusion de ce chapitre est que le modèle de simulation multiniveau proposé dans cette thèse répond à la problématique initiale de nos travaux. Notre modèle permet de

reproduire la simulation d'un système complexe, comme un terminal d'aéroport, tout en gérant efficacement les ressources de calcul disponibles.

III

Conclusion et Perspectives

7

CONCLUSION GÉNÉRALE

7.1/ BILAN

Dans cette thèse, nous avons proposé un modèle et des outils permettant de simuler des piétons dans un environnement intérieur. Une foule de piétons est vue comme un système composé d'un grand nombre d'entités en interaction, dont la dynamique globale ne peut se réduire à la somme des comportements de ses composants. Nous retrouvons dans cette vision toutes les caractéristiques d'un système complexe. Typiquement, ces systèmes exhibent des structures hiérarchiques et des processus d'auto-organisation. La simulation multiniveau fondée sur les modèles multiagents holoniques est l'approche que nous avons choisie pour analyser la dynamique de tels systèmes. Elle permet d'étudier les systèmes en considérant plusieurs niveaux d'observation (microscopique, mésoscopique et macroscopique).

Nous avons appliqué la méthodologie ASPECS, depuis l'analyse jusqu'à l'implantation. Fondée sur le métamodèle CRIO (Capacité, Rôle, Interaction, Organisation), elle nous a fourni un cadre pour la création de modèles holoniques nécessaires à la simulation de foules de piétons. ASPECS adopte une approche dirigée par les modèles (« *Model Driven Development* ») et définit trois domaines, correspondant chacun à une étape du développement de logiciels. Le domaine du problème fournit les abstractions nécessaires pour collecter les connaissances liées au système. Le cœur de ce domaine repose sur les concepts d'organisation et de rôle. L'approche organisationnelle est utilisée pour décomposer un système niveau par niveau. Le domaine agent décrit le modèle d'une société holonique d'agents en charge d'offrir une solution. Le modèle du système à réaliser est instancié et associé à un ensemble de holons en charge de la simulation. Le troisième et dernier domaine est consacré à l'implantation de l'application sur une plateforme spécifique.

Dans cette thèse, nous avons proposé une extension du métamodèle pour chacun de ces domaines. Ces extensions proposent les abstractions nécessaires à la modélisation d'environnements intérieurs, de leurs structures et de leurs dynamiques. L'extension de CRIO dans le domaine du problème propose les abstractions pour modéliser la décomposition structurelle de l'environnement (bâtiment, étage, couloir, salle, *etc.*). Le modèle organisationnel associé permet la modélisation et la simulation des missions classiquement assignées à l'environnement : calcul des perceptions des piétons, gestion des actions des piétons dans l'environnement, *etc.* Les relations entre les organisations de l'environnement permettent de construire une hiérarchie organisationnelle exprimée en termes de holons. Il en résulte un modèle holonique de l'environnement, permettant de faire cohabiter plusieurs niveaux d'abstraction, au sein d'une même simulation. Chaque holon est

situé à un niveau spécifique et est en charge de simuler une zone de l'environnement. Il met en œuvre une évaluation de son comportement basée sur des indicateurs énergétiques. Les différences d'énergie entre deux niveaux adjacents permettent de déterminer si le modèle utilisé au niveau supérieur approxime correctement l'ensemble des comportements des modèles des niveaux inférieurs. L'ensemble de ces évaluations permet de construire dynamiquement la hiérarchie de holons devant simuler le comportement de l'environnement intérieur tout en considérant des critères qualitatifs et quantitatifs.

Afin d'illustrer nos travaux sur une application concrète, nous avons étudié la simulation de foules de voyageurs entre les halls d'enregistrement et d'embarquement d'un aéroport. Cette application a permis de mettre en avant le comportement de nos modèles d'environnement pour de larges espaces ouverts. Par ailleurs, la simulation des postes de contrôle a permis d'illustrer le comportement de nos modèles dans une zone retreinte à forte densité de population. Les indicateurs multiniveaux sont raffinés et détaillés pour leur application à la simulation du terminal de l'aéroport. Nous avons choisi de restreindre ces indicateurs à la mission de génération des perceptions des piétons. En effet, nous considérons cette mission comme celle ayant la plus grande influence sur les comportements des piétons. Cette application a mis en évidence les qualités et les défauts de notre approche.

Les principaux défauts concernent :
- La difficulté à trouver une analogie de l'énergie pour d'autres domaines d'étude (autre que la simulation de foules).
- Le rapport entre la précision des résultats de simulation et ceux fournis par le système réel.
- L'impossibilité de simuler le système en temps réel à partir d'un certain seuil (*cf.* chapitre 6, page 130).

Les principaux avantages des abstractions et des outils présentés dans cette thèse peuvent être résumés comme suit :
- Réutilisation et modularité des modèles et des connaissances, en partie grâce à l'adoption de l'approche organisationnelle.
- Modélisation d'un système d'environnement intérieur à plusieurs niveaux d'abstraction grâce à l'approche holonique.
- Proposition et exploitation d'indicateurs pour évaluer et estimer les écarts entre les différents niveaux de simulation.

Les travaux présentés dans cette thèse nécessitent plusieurs approfondissements. La section suivante recense quelques-unes des améliorations possibles pouvant faire l'objet de futures recherches.

7.2/ PERSPECTIVES

Nous définissons deux axes d'approfondissement pour la suite de ce travail. Le premier s'intéresse à la simulation multiniveau. Le second axe concerne l'extension des métamodèles proposés dans ce manuscrit pour intégrer de l'information sémantique dans la modélisation de l'environnement.

7.2.1/ SIMULATION MULTINIVEAU

Cette thèse a introduit les systèmes multiagents holoniques pour mettre en œuvre la simulation multiniveau de foules de piétons. Des efforts sont nécessaires pour l'extension et la généralisation de ces conclusions en vue d'enrichir la plateforme JANUS par un simulateur multiniveau. Par ailleurs, la création de nouveaux indicateurs et outils pour l'évaluation de la cohérence d'une simulation est nécessaire.

En effet, il est difficile de trouver une analogie énergétique dans d'autres domaines ; et d'autres types d'indicateurs (non physique ou énergétique) pourraient répondre au problème d'évaluation de la précision des résultats de simulation.

L'adaptation des modèles inspirés d'autres domaines, et notamment de la physique, semble prometteuse. La physique statistique offre une palette importante d'outils qui pourront, par analogie, être transposés au domaine des SMA (*e.g.* la fonction Z).

7.2.2/ INTÉGRATION DE LA SÉMANTIQUE DES OBJETS DANS LA MODÉLISATION DE L'ENVIRONNEMENT

Il s'agit d'étendre le modèle de l'environnement en intégrant la sémantique associée aux entités qui le composent. Ces informations supplémentaires peuvent être utiles pour doter les piétons de comportements plus évolués. Ainsi, les agents piétons pourraient mettre en place des plans d'actions en considérant les usages possibles des objets qu'ils perçoivent.

L'impact sur notre modèle serait la possibilité d'intégrer des comportements plus complexes des piétons et donc d'obtenir une plus grande précision dans les résultats de simulation. Toutefois, le modèle d'environnement devrait être couplé à une hiérarchie pour la sémantique des objets. Cette hiérarchie pourrait être utilisée pour regrouper les objets.

De nombreuses applications de la simulation de piétons peuvent bénéficier de ces informations. Citons deux exemples d'application que nous souhaitons aborder au sein de notre laboratoire. Le premier concerne la simulation de l'évacuation d'un bâtiment en feu. Associer une sémantique *« Sortie »* à des objets de l'environnement permettrait aux usagers du bâtiment de déterminer le meilleur chemin d'évacuation, en suivant les directions indiquées par cette signalétique. Un second exemple pourrait être la qualification de l'usage d'un bâtiment. La sémantique associée aux salles et aux objets à l'intérieur du bâtiment permettrait de modéliser des comportements fins d'usagers et ensuite d'évaluer ces comportements pour qualifier le bâtiment face à son usage.

IV

BIBLIOGRAPHIE

BIBLIOGRAPHIE

Frédéric Amblard. *Comprendre le fonctionnement de simulations sociales individus-centrées, Application à des modèles de dynamiques d'opinions*. Thèse de doctorat, Cemagref, Clermont-Ferrand, France, Décembre 2003.

Egil P. Anderson et Trygve Reenskaug. System Design by Composing Structures of Interacting Objects. Dans *European Conference on Object-Oriented Programming*, volume 615 de *Lecture Notes*, pages 133–153. Springer Verlag, 1992.

Nguyen Thi Ngoc Anh, Zucker Jean Daniel, Nguyen Huu Du, Alexis Drogoul et Vo Duc An. A hybrid macro-micro pedestrians evacuation model to speed up simulation in road networks. Dans *Proceedings of the 10th international conference on Advanced Agent Technology*, AAMAS'11, pages 371–383, Berlin, Heidelberg, 2012. Springer-Verlag. ISBN 978-3-642-27215-8.

D.K. Arvind et C.R. Smart. Hierachical parallel discrete event simulation in composite ELSA. Dans *6th Workshop on Parallel and Distributed Simulation (PADS'92)*, pages 147–158, Newport Beach, USA, janvier 1992.

D.K. Arvind, C.R. Smart et V.E. Rebello. Distributed simulation of parallel VLSI architectures. Dans *Workshop on Architecture, Algorithms and VLSI*, pages 285–298, Bonas, France, juin 1991.

Norman Ashford. Level of service design concept for airport passenger terminals—a european view. *Transportation Planning and Technology*, 12(1):5–21, 1988.

T. Balch. Hierarchic social entropy : An information theoretic measure of robot group diversity. Dans *Autonomous Robots*, volume 8, July 2000.

T. Balser, K. McMahon, D. Bart, D. Bronson, D. Coyle, N. Craig, M. Flores-Mangual, K. Forshay, S. Jones, A. Kent et A. Shade. Bridging the gap between micro- and macro-perspectives on the role of microbial communities in global change ecology. *Plant and Soil*, 289:59–70, 2006.

M. Balzer et O. Deussen. Level-of-detail visualization of clustered graph layouts. Dans *Asia-Pacific Symposium on Visualisation (APVIS)*, pages 133–140, Feb 2007.

John S. Baras et Xiaobo Tan. Control of autonomous swarms using gibbs sampling. Dans *43rd IEEE Conf. on Decision and Control*, Bahamas, dec 2004.

Mihai Barbuceanu et Mark S Fox. *COOL : A language for describing coordination in multi agent systems*, volume pages, pages 1–15. Citeseer, 1995.

Momotaz Begum et Fakhreddine Karray. Visual attention for robotic cognition : A survey. *IEEE T. Autonomous Mental Development*, 3(1):92–105, 2011.

Carole Bernon, Maire-Pierre Gleizes, Sylvain Peyruqueou et Gauthier Picard. ADELFE, a methodology for adaptive multi-agent systems engineering. Dans *Third International Workshop Engineering Societies in the Agents World (ESAW)*, volume 2577 de *LNAI*, pages 156–169, Madrid, Spain, September 2002. Springer-Verlag.

Orna Berry. *Performance Evaluation of the Time Warp Distributed Simulation Mechanism*. Thèse de doctorat, University of South California, Los Angeles, USA, janvier 1986.

Victor J. Blue et Jeffrey L. Adler. Cellular automata microsimulation for modeling bi-directional pedestrian walkways. *Transportation Research Part B-methodological*, 35:293–312, 2001.

Barry W. Boehm. A spiral model of software development and enhancement. *Computer*, 21(5):61–72, 1988. ISSN 0018-9162.

Olivier Boissier et Yves Demazeau. Asic : An architecture for social and individual control and its application to computer vision. Dans *Preproceedings of the 6th European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, pages 107–118, 1994.

Rafael H. Bordini, Antônio Carlos da Rocha Costa, Jomi F. Hübner, Álvaro F. Moreira, Fabio Y. Okuyama et Renata Vieira. MAS-SOC : a social simulation platform based on agent-oriented programming. *Journal of Artificial Societies and Social Simulation*, 8(3), 2005.

Leonardo Bottaci. A direct manipulation interface for a user enhanceable crowd simulator. *Journal of Intelligent Systems*, 5(2-4):249–272, 1995. ISSN 0334-1860.

B.D. Bowes. *The Effects of Emotion and Time to Shop on Shopping Behaviour in an International Airport Terminal*. Griffith University, 1999.

A. Braun, B.J. Bodman et S.R. Musse. Simulating virtual crowds in emergency situations. Dans *ACM Symposium on Virtual Reality Software and Technology*, pages 244–252, Monterey, CA, USA, 2005a. ACM.

Adriana Braun, Bardo E. J. Bodmann et Soraia R. Musse. Simulating virtual crowds in emergency situations. Dans *Proceedings of the ACM symposium on Virtual reality software and technology*, VRST '05, pages 244–252, New York, NY, USA, 2005b. ACM. ISBN 1-59593-098-1.

Anne Bretagnolle, Eric Daudé et Denise Pumain. From theory to modelling : urban systems as complex systems. *Cybergeo, 13th European Colloquium on Quantitative and Theoretical Geography*, 335, September 2003.

Jean-Pierre Briot et Yves Demazeau. *Introduction aux systèmes multi-agents*, pages 17–25. Collection IC2. Hermès-Lavoisier, 2001. ISBN 2-7462-0336-7.

David C. Brogan et Jessica K. Hodgins. Simulation level of detail for multiagent control. Dans *AAMAS '02 : Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 199–206, New York, NY, USA, 2002. ACM. ISBN 1-58113-480-0.

Christopher H. Brooks, Edmund H. Durfee et Aaron A. Armstrong. An introduction to congregating in multiagent systems. Dans *4th International Conference on Multi-Agent Systems (ICMAS 2000), 10-12 July 2000, Boston, MA, USA*, pages 79–86. IEEE Computer Society, 2000. ISBN 0-7695-0625-9.

Randall Bryant. Simulation on a distributed system. Dans *First International Conference on Distributed Computing Systems (ICDCS)*, pages 544–552, Alabama, USA, octobre 1979.

Wilco Burghout. *Hybrid microscopic-mesoscopic traffic simulation*. Thèse de doctorat, KTH Infrastructure, Royal Institute of Technology, Stockholm, Sweden, dec 2004.

Wilco Burghout, H. Koutsopoulos et I. Andréasson. Hybrid mesoscopic-microscopic traffic simulation. Dans *World Conference on Transportation Research CTR2004*, volume 02, 2004.

Wilco Burghout, Haris N. Koutsopoulos et Ingmar Andréasson. Hybrid mesoscopic-microscopic traffic simulation. *Transportation Research Record : Journal of the Transportation Research Board*, 1934:218–255, 2005.

C Burstedde, K Klauck, A Schadschneider et J Zittartz. Simulation of pedestrian dynamics using a two-dimensional cellular automaton. *Physica A : Statistical Mechanics and its Applications*, 295(3-4):507 – 525, 2001. ISSN 0378-4371.

Giovanni Caire, Wim Coulier, Francisco J. Garijo, Jorge Gomez, Juan Pavón, Francisco Leal, Paulo Chainho, Paul E. Kearney, Jamie Stark, Richard Evans et Philippe Massonet. Agent Oriented Analysis Using Message/UML. Dans Michael Wooldridge, Gerhard Weiß et Paolo Ciancarini, éditeurs, *Agent-Oriented Software Engineering II, Second International Workshop, AOSE 2001, Montreal, Canada, May 29, 2001, Revised Papers and Invited Contributions*, volume 2222 de *Lecture Notes in Computer Science*, pages 119–135. Springer Verlag, 2002. ISBN 3-540-43282-5.

Benjamin Camus, Julien Siebert, Christine Bourjot et Vincent Chevrier. Modélisation multi-niveaux dans AA4MM. Dans Pierre Chevailler et Bruno Mermet, éditeurs, *Journées Francophones sur les Systèmes Multi-Agents*, pages 43–52, Honfleur, France, octobre 2012. Cépaduès.

L. Cernuzzi, M. Cossentino et F. Zambonelli. Process models for agent-based development. *Journal of Engineering Applications of Artificial Intelligence (EAAI)*, 18(2), March 2005.

B. Chaib-Draa, I. Jarras et B. Moulin. Systèmes multi-agents : principes généraux et application. Dans Jean P. Briot et Yves Demazeau, éditeurs, *Principes et architecture des systèmes multi-agents*, chapitre 1. Hermès-Lavoisier, 2001.

K. Mani Chandy et J. Misra. Asynchronous distributed simulation via a sequence of parallel computations. *Communications of the ACM*, 24(11):198–206, avril 1981.

K.M. Chandy et J. Misra. Distributed simulation : A case study in design and verification of distributed programs. *IEEE Transaction on Software Engineering*, 5(5):440–452, 1979.

K.M. Chandy et J. Misra. *Parallel Program Design : a Foundation*. Addison-Wesley, 1988.

A. Chella, M. Cossentino, L. Sabatucci et V. Seidita. Agile PASSI : An agile process for designing agents. *International Journal of Computer Systems Science and Engineering. Special issue on Software Engineering for Multi-Agent Systems*, 21(2), March 2006.

Stephen Chenney, Okan Arikan et D. A. Forsyth. Proxy simulations for efficient dynamics. Dans *Eurographics, Short Presentations*, 2001.

Stephen Chenney et David Forsyth. View-dependent culling of dynamic systems in virtual environments. Dans *Proceedings of the symposium on Interactive 3D graphics (SI3D)*, pages 55–58, New York, NY, USA, April 1997. ACM Press. ISBN 0-89791-884-3.

Mohcine Chraibi, Armin Seyfried et Andreas Schadschneider. Generalized centrifugal-force model for pedestrian dynamics. *Phys Rev E Stat Nonlin Soft Matter Phys*, 82(4 Pt 2):046111, 2010. ISSN 1550-2376.

R. Conte, N. Gilbert et J. Simao Sichman. MAS and social simulation : A suitable commitment. Dans *First International Workshop on Multi-Agent Systems and Agent-Based Simulation*, volume 1534 de *LNCS*, pages 1–9. Springer-Verlag, 1998.

J.M. Contet, F. Gechter, P. Gruer et A. Koukam. Physics inspired multiagent model for vehicle platooning. Dans *International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, May 2007.

Patrick Coquillard et David R.C. Hill. *Modélisation et Simulation des Ecosystèmes*. Masson, 1997.

Kelly Christine Correa e Silva Fernandes. *Systèmes Multi-Agents Hybrides : Une Approche pour la Conception de Systèmes Complexes*. Thèse de doctorat, Université Joseph Fourier- Grenoble 1, 2001.

Massimo Cossentino. From Requirements to Code with the PASSI Methodology. Dans B. Henderson-Sellers et P. Giorgini, éditeurs, *Agent-Oriented Methodologies*, chapitre IV, pages 79–106. Idea Group Publishing, Hershey, PA, USA, 2005.

Massimo Cossentino, Nicolas Gaud, Vincent Hilaire, Stéphane Galland et Abderrafiaa Koukam. ASPECS : an agent-oriented software process for engineering complex systems - how to design agent societies under a holonic perspective. *Autonomous Agents and Multi-Agent Systems*, 2(2):260–304, mar 2010.

D. Curcio, F. Longo, G. Mirabelli et E. Papoff. Passengers' flow analysis and security issues in airport terminals using modeling & simulation. Dans *European Conference on Modeling & Simulation*, Praga-Repubblica Ceca, 4-6 June 2007.

Luiz Gonzaga da Silveira et Soraia Raupp Musse. Real-time generation of populated virtual cities. Dans *Proceedings of the ACM symposium on Virtual reality software and technology*, VRST '06, pages 155–164, New York, NY, USA, 2006a. ACM. ISBN 1-59593-321-2.

Luiz Gonzaga da Silveira et Soraia Raupp Musse. Real-time generation of populated virtual cities. Dans *Proceedings of the ACM symposium on Virtual reality software and technology*, VRST '06, pages 155–164, New York, NY, USA, 2006b. ACM. ISBN 1-59593-321-2.

Winnie Daamen. *Modelling Passenger Flows in Public Transport Facilities*. Thèse de doctorat, Technical University of Delft, 2004.

Paul Davidsson. Multi agent based simulation : Beyond social simulation. Dans Scott Moss et Paul Davidsson, éditeurs, *Multi-Agent-Based Simulation*, volume 1979 de *Lecture Notes in Computer Science*, pages 141–155. Springer Berlin / Heidelberg, 2001. ISBN 978-3-540-41522-0.

Percival Silva de Lucena, Carlos Alberto Estombelo Montesco et Dilvan de Abreu Moreira. Semanticagent, a platform for the development of software agents. Dans *1 Workshop em Tecnologia da Informação e da Linguagem Humana*, volume 1, pages 1–4, 2003. 12 de outubro.

Scott A. DeLoach, Mark F. Wood et Clint H. Sparkman. Multiagent systems engineering. *The International Journal of Software Engineering and Knowledge Engineering*, 11(3), June 2001.

Edsger W. Dijkstra et Carel S. Scholten. Termination detection of diffusing computation. *Informations Processing Letter*, (11):217–219, août 1980.

Jan Dijkstra et Harry Timmermans. Towards a multi-agent model for visualizing simulated user behavior to support the assessment of design performance. *Automation in Construction*, 11(2):135 – 145, 2002. ISSN 0926-5805.

Stephane Donikian. Multilevel modelling of virtual urban environments for behavioural animation. Dans *CA '97 : Proceedings of the Computer Animation*, pages 127–133, Washington, DC, USA, 1997a. IEEE Computer Society. ISBN 0-8186-7984-0.

Stephane Donikian. VUEMS : A virtual urban environment modeling system. *Computer Graphics International Conference*, 0:84, 1997b.

Kurt Dopfer, John Foster et Jason Potts. Micro-meso-macro. *Journal of Evolutionary Economics*, 14(3): 263–279, July 2004.

A. Drogoul. *De la simulation multi-agents à la résolution collective de problèmes*. Thèse de doctorat, University Paris 6, Paris, France, November 1993.

A. Drogoul et S. Picault. Microbes : Vers des collectivités de robots socialement situés. Dans M.-P. Gleizes et P. Marcenac, éditeurs, *VIIèmes Journées Francophones pour l'Intelligence Artificielle Distribuée et les Systèmes Multi-Agents (JFIADSMA)*, pages 265–278. Hermès, 1999.

R. Ducrot, A. Botta, P. D'Aquino, M. Antona, G. Abrami, S. Farolfi, J.P. Müller, E. Lagabrielle et C. Le Page. Changement d'échelle et niveaux d'organisation multiples. Dans *La modélisation d'accompagnement : une démarche participative en appui au développement durable*, pages 251–275. Editions Quae, 2010. ISBN 9782759206209.

P. Eades et Q. W. Feng. Multilevel visualization of clustered graphs. Dans S. North, éditeur, *Graph Drawing 96*, volume 1190 de *Lecture Notes in Computer Science*. Springer, 1996.

A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22:46–57, June 1989. ISSN 0018-9162.

Ronald Fagin. *Reasoning about Knowledge*. MIT Press, 1995.

Brett R. Fajen, William H. Warren, Selim Temizer et Leslie Pack Kaelbling. A dynamical model of visually-guided steering, obstacle avoidance, and route selection. *Int. J. Comput. Vision*, 54(1-3):13–34, août 2003. ISSN 0920-5691.

Nathalie Farenc. *An informed environment for inhabited city simulation*. Thèse de doctorat, Lausanne, 2001.

Nathalie Farenc, Ronan Boulic et Daniel Thalmann. An informed environment dedicated to the simulation of virtual humans in urban context. *Computer Graphics Forum*, 18(3):309–318, 1999.

I Farkas, D Helbing et T Vicsek. Mexican waves in an excitable medium. *Nature*, 419(6903):131–2, 2002. ISSN 0028-0836.

J. Ferber, O. Gutknecht et F. Michel. From agents to organizations : an organizational view of multi-agent systems. Dans *Agent-Oriented Software Engineering IV 4th International Workshop*, volume 2935 de *LNCS*, pages 214–230, Melbourne, Australia, mar 2004. Springer Verlag.

J. Ferber, F. Michel et J. Baez. AGRE : Integrating environments with organizations. Dans Springer Berlin / Heidelberg, éditeur, *Environments for Multi-Agent Systems (E4MAS)*, volume 3374, pages 48–56, 2005.

J Ferber et J.-P. Müller. Influences and reactions : a model of situated multiagent systems. Dans AAAI Press, éditeur, *ICMAS'96 (International Conference on Multi-Agent Systems)*, 1996.

Jacques Ferber. *Les systèmes multi-agents. Vers une intelligence collective.* 1995.

Jacques Ferber. *Multi-Agent Systems : An Introduction to Distributed Artificial Intelligence*. Addison-Wesley Professional, février 1999. ISBN 0201360489.

Jacques Ferber, Fabien Michel et Olivier Gutknecht. Agent/group/roles : Simulating with organizations. Dans Jean-Pierre Müller, éditeur, *Agent-Based Simulation 4*, 28-30 April 2003.

J.M. Filloque. *Synchronisation répartie sur une machine parallèle à couche logique reconfigurable*. Thèse de doctorat, Université de Rennes 1, 1992.

Tim Finin, Richard Fritzson, Don Mckay et Robin Mcentire. Kqml as an agent communication language. MIT Press, 1994.

FIPA. Agent communication language. Rapport technique, 1998.

Paul A. Fishwick. Computer simulation : growth through extension. *Trans. Soc. Comput. Simul. Int.*, 14(1): 13–23, 1997. ISSN 0740-6797.

FIPA ACL, 2002. *FIPA ACL Message Structure Specification*. Foundation For Intelligent Physical Agents, 2002. Standard, SC00061G.

FIPA Com. Act, 2002. *FIPA Communicative Act Library Specification*. Foundation For Intelligent Physical Agents, 2002. Standard, SC00037J.

Alfonso Fuggetta. Software process : a roadmap. Dans ACM Press, éditeur, *Int. Conference on Software Engineering (ICSE), Future of Software Engineering Track*, pages 25–34, Limerick, Ireland, June 2000.

Richard Fujimoto. Optimistic approachs to parallel discrete event simulation. *Transactions of the Society for Computer Simulation*, 7(2):152–191, juin 1990a.

Richard Fujimoto. Parallel discrete event simulation. Dans *ACM*. octobre 1990b.

Minoru Fukui et Yoshihiro Ishibashi. Jamming Transition in Cellular Automaton Models for Pedestrians on Passageway. *Journal of the Physical Society of Japan*, 68(11):3738–3739, 1999a.

Minoru Fukui et Yoshihiro Ishibashi. Self-Organized Phase Transitions in Cellular Automaton Models for Pedestrians. *Journal of the Physical Society of Japan*, 68(8):2861–2863, 1999b.

Anat Gafni. Roolback mechanisms for optimistic distributed simulation systems. Dans *SCS Multi-Conference on Distributed Simulation*, pages 61–67, San Diego, USA, février 1988.

Stéphane Galland, Nicolas Gaud, Jonathan Demange et Abderrafiaa Koukam. Environment model for multiagent-based simulation of 3D urban systems. Dans *the 7th European Workshop on Multiagent Systems (EUMAS09)*, Ayia Napa, Cyprus, dec 2009.

Nicolas Gaud. *Systèmes multi-agents holoniques : de l'analyse à l'implantation. Méta-modèle, méthodologie, et simulation multi-niveaux*. Thèse de doctorat, Université de Franche-Comté et Univérsité de technologie de Belfort-Montbéliard, 2007.

Nicolas Gaud, Stéphane Galland, Vincent Hilaire et Abderrafiaa Koukam. An organizational platform for holonic and multiagent systems. Dans *the Sixth International Workshop on Programming Multi-Agent Systems (ProMAS'08), of the Seventh International Conference on Autonomous agents and Multiagent Systems (AAMAS)*, pages 111–126, Estoril, Portugal, 2008a.

Nicolas Gaud, Stéphane Galland et Abderrafiâa Koukam. Towards a multilevel simulation approach based on holonic multiagent systems. Dans *UKSIM '08 : Proceedings of the Tenth International Conference on Computer Modeling and Simulation*, pages 180–185, Washington, DC, USA, 2008b. IEEE Computer Society. ISBN 978-0-7695-3114-4.

Nicolas Gaud, Franck Gechter, Stéphane Galland et Abderrafiaa Koukam. Holonic multiagent multilevel simulation application to real-time pedestrians simulation in urban environment. Dans *IJCAI07*, pages 1275–1280. 30th International Joint Conference on Artificial Intelligence, January 2007.

Michael R Genesereth. *An Agent-Based Framework for Interoperability*, pages 317–346. AAAI Press / The MIT Press, 1997.

Carlo Ghezzi, Mehdi Jazayeri et Dino Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, second édition, 2002. ISBN 0-13-305699-6.

S. Ghosh. On the concept of dynamic multi-level simulation. Dans *the 19th Annual Symposium on Simulation*, pages 201–205, Tampa, Florida, U.S.A, 1986. ISBN 0-8186-0715-7.

J. J. Gibson. Visually controlled locomotion and visual orientation in animals. *British Journal of Psychology*, 49(3):182–194, août 1958. ISSN 0007-1269.

Nigel Gilbert et Klaus G. Troitzsch. *Simulation for the Social Scientist*. Open University Press, Maidenhead and New York, 2 édition, 2005.

Christian Gloor, Pascal Stucki et Kai Nagel. Hybrid techniques for pedestrian simulations. Dans 4^{th} *Swiss Transport Research Conference STRC*, Monte Verità, Ascona, March 2004.

Kasper Bilsted Graversen. *The nature of roles, A taxonomic analysis of roles as a language construct*. Thèse de doctorat, IT University of Copenhagen, Denmark, 2006.

Nathan Griffiths. Supporting Cooperation Through Clans. Dans *Proceedings of the 2nd IEEE Systems, Man and Cybernetics, UK & RI Chapter Conference*, 2003.

Thomas Gruber. Toward principles for the design of ontologies used for knowledge sharing. *International Journal Human-Computer Studies*, 43(Issues 5-6):907–928, November 1995.

Mahdi Hannoun, Olivier Boissier, Jaime Simao Sichman et Claudette Sayettat. MOISE : An organizational model for multi-agent systems. Dans Monard M. et Sichman J., éditeur, *Advances in Artificial Intelligence, IBERAMIA-SBIA*, pages 156–165, Brazil, 2000.

P. Hareesh. Evacuation simulation : Visualisation using virtual humans in a distributed multi-user immersive vr system. Dans *VSMM'00*, 2000.

Sandra C. Hayden, Christian Carrick et Qiang Yang. A catalog of agent coordination patterns. Dans *Proceedings of the third annual conference on Autonomous Agents*, AGENTS '99, pages 412–413, New York, NY, USA, 1999. ACM. ISBN 1-58113-066-X.

D. Helbing. A Fluid Dynamic Model for the Movement of Pedestrians. *eprint arXiv :cond-mat/9805213*, mai 1998.

Dirk Helbing, Illes Farkas et Tamas Vicsek. Simulating Dynamical Features of Escape Panic. *Nature*, 407: 487–490, septembre 2000.

Dirk Helbing et Peter Molnar. Social force model for pedestrian dynamics. *PHYSICAL REVIEW E*, 51: 4282–4286, 1995.

Dirk Helbing et Martin Treiber. Gas-kinetic-based traffic model explaining observed hysteretic phase transition. *Phys. Rev. Lett.*, 81(14):3042–3045, Oct. 1998.

H. Hexmoor et G. Beavers. Towards teams of agents. Dans CSREA Press, éditeur, *Proceedings of the International Conference in Artificial Intelligence*, IC-AI'2001, 2001.

Vincent Hilaire, Abder Koukam, Pablo Gruer et Jean-Pierre Müller. Formal specification and prototyping of multi-agent systems. Dans Andrea Omicini, Robert Tolksdorf et Franco Zambonelli, éditeurs, *Engineering Societies in the Agents' World*, numbéro 1972 dans Lecture Notes in Artificial Intelligence. Springer Verlag, 2000.

John H. Holland. *Hidden order : how adaptation builds complexity*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, july 1995. ISBN 0-201-40793-0.

Serge Hoogendoorn. Pedestrian Flow Modeling by Adaptive Control. *Transportation Research Record*, 1878: 95–103, 2004.

Serge Hoogendoorn et Piet H. L. Bovy. Simulation of pedestrian flows by optimal control and differential games. *Optimal Control Applications & Methods*, 24:153–172, 2003.

Serge P. Hoogendoorn et Piet H.L. Bovy. State-of-the-art of vehicular traffic flow modelling. *Journal of Systems and Control Engineering*, 215(4):283–303, aug 2001.

Bryan Horling et Victor Lesser. A survey of multi-agent organizational paradigms. *Knowledge Engineering Review*, 19(4):281–316, 2004. ISSN 0269-8889.

Wesley H. Huang, Brett R. Fajen, Jonathan R. Fink et William H. Warren. Visual navigation and obstacle avoidance using a steering potential function. *Robotics and Autonomous Systems*, 54(4):288–299, 2006.

R. Hughes. A continuum theory for the flow of pedestrians. *Transportation Research Part B : Methodological*, 36(6):507–535, juillet 2002. ISSN 01912615.

Patrice Ingels et Michel Raynal. Simulation répartie de systèmes à événements discrets. *TSI*, 9(5):383–397, octobre 1990.

David Isern, David Sánchez et Antonio Moreno. Organizational structures supported by agent-oriented methodologies. *J. Syst. Softw.*, 84:169–184, February 2011. ISSN 0164-1212.

ISO-3534-1. Statistique — vocabulaire et symboles — partie 1 : Probabilité et termes statistiques généraux et termes utilisés en calcul des probabilités. Rapport technique, 2006.

Rakiatou Christelle Jackou. *Contribution à la Gestion des Opérations de la Sûreté Aéroportuaire : Modélisation et Optimisation*. Thèse de doctorat, Institut National Polytechnique de Toulouse, Toulouse, France, novembre 2010.

W. Jager, R. Popping et H. van de Sande. Clustering and fighting in two-party crowds : Simulating the approach-avoidance conflict. *Journal of Artificial Societies and Social Simulation*, 4(3), 2001.

Imed Jarras et Brahim Chaib-draa. Aperçu sur les systèmes multiagents. 2002.

D. Jefferson. Virtual time. *ACM Transactions on Programming Languages and Systems*, 7(3):404–425, 1985.

David Jefferson et Henry Sowizral. Fast concurrent simulation using the time warp mechanism. Dans *SCS Multi-Conference on Distributed Simulation*, pages 63–69, San Diego, USA, janvier 1985.

Nicholas R. Jennings. On agent-based software engineering. *Artif. Intell.*, 117:277–296, March 2000. ISSN 0004-3702.

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

Li Jian, Yang Lizhong et Zhao Daoliang. Simulation of bi-direction pedestrian movement in corridor. *Physica A : Statistical Mechanics and its Applications*, 354(0):619 – 628, 2005. ISSN 0378-4371.

Hao Jiang, Wenbin Xu, Tianlu Mao, Chunpeng Li, Shihong Xia et Zhaoqi Wang. Continuum crowd simulation in complex environments. *Computers & Graphics*, 34(5):537 – 544, 2010. ISSN 0097-8493.

A Johansson, D Helbing et P Shukla. Specification of the social force pedestrian model by evolutionary adjustment to video tracking data. *Advances in Complex Systems*, 10(supp02):271–288, 2007.

Paul E. Joustra et Nico M. Van Dijk. Simulation of check-in at airports. Dans *Proceedings of the 33nd conference on Winter simulation*, WSC '01, pages 1023–1028, Washington, DC, USA, 2001. IEEE Computer Society. ISBN 0-7803-7309-X.

Marcelo Kallmann, Hanspeter Bieri et Daniel Thalmann. Fully dynamic constrained delaunay triangulations. Dans G. Brunnett, B. Hamann, H. Mueller et L. Linsen, éditeurs, *Geometric Modeling for Scientific Visualization*, pages 241–257. Springer-Verlag, Heidelberg, Germany, 2003. ISBN 3-540-40116-4.

R. Kemp et J. Van den Bergh. Economics and transitions : Lessons from economic sub-disciplines. UNU-MERIT Working Paper Series 038, United Nations University, Maastricht Economic and social Research and training centre on Innovation and Technology, 2006.

David Kinny, Michael Georgeff et Anand Rao. A methodology and modelling technique for systems of BDI agents. Dans Rudy van Hoe, éditeur, *Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, Eindhoven, The Netherlands, 1996.

Ansgar Kirchner, Hubert Klüpfel, Katsuhiro Nishinari, Andreas Schadschneider et Michael Schreckenberg. Discretization effects and the influence of walking speed in cellular automata models for pedestrian dynamics. *Journal of Statistical Mechanics : Theory and Experiment*, 2004(10):P10011, 2004.

Ansgar Kirchner et Andreas Schadschneider. Simulation of evacuation processes using a bionics-inspired cellular automaton model for pedestrian dynamics. *Physica A : Statistical Mechanics and its Applications*, 312(1–2):260 – 276, 2002. ISSN 0378-4371.

Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda et Eiichi Osawa. RoboCup : The robot world cup initiative. Dans W. Lewis Johnson et Barbara Hayes-Roth, éditeurs, *Proceedings of the First International Conference on Autonomous Agents (Agents)*, pages 340–347, New York, 1997. ACM Press. ISBN 0-89791-877-0.

H. Klüpfel, M. Meyer-König, J. Wahle et M. Schreckenberg. Microscopic simulation of evacuation processes on passenger ships. *Theoretical and Practical Issues on Cellular Automata*, pages 63–71, 2000.

Matthias Klusch et Andreas Gerber. Dynamic coalition formation among rational agents. *IEEE Intelligent Systems*, 17:42–47, May 2002. ISSN 1541-1672.

Arthur Koestler. *The Ghost in the Machine*. Hutchinson, 1967.

B.B. Kristensen et K. Osterbye. Roles : Conceptual abstraction theory and practical language issues. *Theory and Practice of Object Systems (TAPOS), Special Issue on Subjectivity in Object-Oriented Systems*, 2(3): 143–160, 1996.

F. Lamarche et S. Donikian. Crowds of Virtual Humans : a New Approach for Real Time Navigation in Complex and Structured Environments. *Computer Graphics Forum (Proc. of Eurographics 2004)*, 23(3), 2004.

Oliver Lamotte, Stephane Galland, Jean-Michel Contet et Franck Gechter. Submicroscopic and physics simulation of autonomous and intelligent vehicles in virtual reality. *Advances in System Simulation, International Conference on*, 0:28–33, 2010.

H Van Landeghem et A Beuselinck. Reducing passenger boarding time in airplanes : A simulation based approach. *European Journal of Operational Research*, 142(2):294 – 308, 2002. ISSN 0377-2217.

Samuel Lemercier. *Simulation du comportement de suivi dans une foule de piétons à travers l'expérience, l'analyse et la modélisation*. These, Université Rennes 1, avril 2012.

Julien Lepagnot et Guillaume Hutzler. A multiscale agent-based model for the simulation of avascular tumour growth. *Journal of Biological Physics and Chemistry*, 9(1):17–25, janvier 2009.

J. Li, W. Ge, J. Zhang et M. Kwauk. Multi-scale compromise and multi-level correlation in complex systems. Dans *A6 Special Issue : 7th World Congress of Chemical Engineering*, volume 83 de *Chemical Engineering Research and Design (ChERD)*, pages 574–582. Institution of Chemical Engineers, June 2005.

Chunxia LU. Analysis of compressed force in crowds. *Journal of Transportation Systems Engineering and Information Technology*, 7(2):98 – 102, 2007. ISSN 1570-6672.

Boris Lubachevsky, Adam Shwartz et Alan Weiss. Roolback sometimes works ... if filtered. Dans *1989 Winter Simulation Conference*, pages 630–639, Washington, USA, décembre 1989.

David Luebke et Chris Georges. Portals and mirrors : Simple, fast evaluation of potentially visible sets. Dans *SI3D '95 : Proceedings of the 1995 symposium on Interactive 3D graphics*, pages 105–ff., New York, NY, USA, 1995. ACM. ISBN 0-89791-736-7.

B. I. Lundqvist, A. Bogicevic, S. Dudiy, P. Hyldgaard, S. Ovesson, C. Ruberto, E. Schröder et G. Wahnström. Bridging between micro- and macroscales of materials by mesoscopic models. *Computational Materials Science*, 24(1), 2002.

Rainald Löhner. On the modeling of pedestrian motion. *Applied Mathematical Modelling*, 34(2):366 – 382, 2010. ISSN 0307-904X.

M. R. Macedomia, M. J. Zyda, D. R. Pratt, D. P. Brutzman et P. T. Barham. Exploiting reality with multicast groups : a network architecture for large-scale virtual environments. Dans *Proceedings of the Virtual Reality Annual International Symposium (VRAIS'95)*, VRAIS '95, pages 2–, Washington, DC, USA, 1995. IEEE Computer Society. ISBN 0-8186-7084-3.

Vijay Madisetti, Jean Walrand et David Messerschmitt. WOLF : A roolback algorithm for optimistic distributed simulation. Dans *1988 Winter Simulation Conference*, pages 296–305, San Diego, USA, décembre 1988.

Pattie Maes. Artificial life meets entertainment : lifelike autonomous agents. *Commun. ACM*, 38:108–114, November 1995. ISSN 0001-0782.

L. Magne, S. Rabut et J-F Gabard. Towards an hybrid macro-micro traffic flow simulation model. Dans *Proceedings of the INFORMS Salt Lake City String 2000 Conference*, 2000.

S. Maniccam. Effects of back step and update rule on congestion of mobile objects. *Physica A : Statistical Mechanics and its Applications*, 346(3–4):631 – 650, 2005. ISSN 0378-4371.

Katalin Martinas. Neumannian economy in multi-agent approach. investigation of stability and instability in economic growth. *Interdisciplinary Description of Complex Systems*, 2(1):70–78, 2004.

P. Mathieu, J. C. Routier et Y. Secq. Dynamic organization of multi-agent systems. Dans *Proceedings of the first international joint conference on Autonomous agents and multiagent systems : part 1*, AAMAS '02, pages 451–452, New York, NY, USA, 2002. ACM. ISBN 1-58113-480-0.

Philippe Mathieu et Jean-Christophe Routier. Une contribution du multi-agent aux applications de travail coopératif. *Réseaux et systèmes répartis, calculateurs Parallèles*, 13(2-3):207–226, 2001. Numéro spécial télé-applications.

Bertrand Maury et Juliette Venel. Handling of contacts in crowd motion simulations. Dans Cécile Appert-Rolland, François Chevoir, Philippe Gondret, Sylvain Lassarre, Jean-Patrick Lebacque et Michael Schreckenberg, éditeurs, *Traffic and Granular Flow '07*, pages 171–180. Springer Berlin Heidelberg, 2009. ISBN 978-3-540-77073-2.

Laura A. McLay, Sheldon H. Jacobson et Alexander G. Nikolaev. A sequential stochastic passenger screening problem for aviation security. *IIE Transactions*, 41(6):575–591, 2009.

Horst Mehl. Speed-up of conservative discrete event simulation methods by speculative computing. Dans *SCS Multi-Conference on Avances in Parallel and Dsitributed Simualtion*, pages 163–166, Anaheim, USA, janvier 1991.

Carlos Merida-Campos et Steven Willmott. Modelling coalition formation over time for iterative coalition games. Dans *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '04, pages 572–579, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 1-58113-864-4.

Fabien Michel. Le modèle influence/réaction pour la simulation multi-agents. Dans *Premières Journées Francophones Modèles Formels de l'Interaction (MFI)*, Toulouse, France, Mai 2001.

Fabien Michel. *Formalism, tools and methodological elements for the modeling and simulation of multi-agents systems*. Thèse de doctorat, LIRMM, Montpellier, France, décembre 2004.

Fabien Michel. Le modèle irm4s : le principe influence/réaction pour la simulation de systèmes multi-agents. Dans *Journées Francophones sur les Systèmes Multi-Agents (JFSMA)*, 2006.

Fabien Michel. The IRM4S model : the influence/reaction principle for multiagent based simulation. Dans *AAMAS '07 : Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, pages 1–3, New York, NY, USA, 2007. ACM. ISBN 978-81-904262-7-5.

D. C. Montgomery et E. H. Durfee. Search Reduction in Hierarchical Distributed Problem Solving. *Group Decision and Negotiation*, 2:301–317, 1993.

Gildas Morvan, Daniel Jolly et Damien Charabidze. Thermoregulation in P. Terraenovae aggregations, an agent-based approach. décembre 2007.

Mehdi Moussaïd. *Etude expérimentale et modélisation des déplacements collectifs de piétons*. Thèse de doctorat, Université de Toulouse, Université Toulouse III - Paul Sabatier, June 2010.

S. R. Musse et D. Thalmann. A model of human crowd behavior : Group inter-relationship and collision detection analysis. Dans *Workshop Computer Animation and Simulation of Eurographics*, pages 39–52, 1997.

S. Raupp Musse et D. Thalmann. Hierarchical model for real time simulation of virtual human crowds. Dans *IEEE Trans. on Visualization and Computer Graphics*, volume 7, pages 152–164, 2001.

Ryoichi Nagai et Takashi Nagatani. Jamming transition in counter flow of slender particles on square lattice. *Physica A : Statistical Mechanics and its Applications*, 366(0):503 – 512, 2006. ISSN 0378-4371.

A. Nakayama, Y. Sugiyama et K. Hasebe. Instability of pedestrian flow in two-dimensional optimal velocity model. Dans Nathalie Waldau, Peter Gattermann, Hermann Knoflacher et Michael Schreckenberg, éditeurs, *Pedestrian and Evacuation Dynamics 2005*, pages 321–332. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-47064-9.

Victor Noel. *Component-based Software Architectures and Multi-Agent Systems : Mutual and Complementary Contributions for Supporting Software Development*. Thèse de doctorat, Université de Toulouse, Toulouse, France, Juillet 2012.

Hansrudi Noser, Olivier Renault, Daniel Thalmann et Nadia Magnenat Thalmann. Navigation for digital actors based on synthetic vision, memory and learning. *Computers and Graphics*, 19:7–19, 1995.

MDA, 2003. *MDA Guide, v1.0.1, OMG/2003-06-01*. Object Management Group (OMG), June 2003.

James Odell. Agents and complex systems. *Journal of Object Technology*, 1(2):35–45, July-August 2002.

James Odell, Marian Nodine et Renato Levy. A metamodel for agents, roles, and groups. Dans James Odell, P. Giorgini et Jörg Müller, éditeurs, *Agent-Oriented Software Engineering (AOSE) IV*, Lecture Notes on Computer Science. Springer, 2005.

James J. Odell, H. Van Dyke Parunak, Mitch Fleischer et Sven Brueckner. Modeling agents and their environment. Dans *Proceedings of the 3rd international conference on Agent-oriented software engineering III*, AOSE'02, pages 16–31, Berlin, Heidelberg, 2002. Springer-Verlag. ISBN 3-540-00713-X.

Anthony Kent Ogenyi Omar. International airport influences on impulsive shopping : trait and normative approach. *International Journal of Retail & Distribution Management*, 29:226 – 235, 2001.

S. Okazaki et S. Matshushita. A study of simulation model for pedestrian movement with evcuation and queuing. Dans *International Conference on Engineering for Crowd Safety*, 1993.

Shigeyuki Okazaki et Satoshi Matsushita. A study of simulation model for way finding behavior by experiments in mazes. *Journal of architecture, planning and environmental engineering. Transactions of AIJ*, 429:51–59, 1991.

Jan Ondřej, Julien Pettré, Anne-Hélène Olivier et Stéphane Donikian. A synthetic-vision based steering approach for crowd simulation. *ACM Trans. Graph.*, 29(4):123 :1–123 :9, juillet 2010. ISSN 0730-0301.

M. Owen, E.R. Galea, Lawrence P. et L. Filippidis. The numerical simulation of aircraft evacuation and its application to aircraft design and certification. *Aeronautical Journal*, 102:301–312, 1998.

Q Feng P. Eades. Multilevel visualition of clustered graphs. Technical report 96-09, University of NewCastle, Australia, Sept 1996.

Sébastien Paris, Stéphane Donikian et Nicolas Bonvalet. Vers une architecture pour la simulation microscopique de foule. *REFIG, Revue Electronique d'Informatique Graphique*, 1:33–43, 2007.

H. Van Dyke Parunak, Sven Brueckner et John Sauter. Digital pheromone mechanisms for coordination of unmanned vehicles. Dans *First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 449–450, New York, NY, USA, 2002. ACM Press. ISBN 1-58113-480-0.

H.V.D Parunak et S. Brueckner. Entropy and self-organization in multi-agent systems. Dans *Autonomous Agents*, pages 124–130, 2001.

Stefano Pellegrini, Andreas Ess, Konrad Schindler et Luc J. Van Gool. You'll never walk alone : Modeling social behavior for multi-target tracking. Dans *ICCV'09*, pages 261–268, 2009.

G.C. Pettinaro, I.W. Kwee et L.M. Gambardella. Acceleration of 3D dynamics simulation of s-bot mobile robots using multi-level model switching. Rapport technique IDSIA-20-03, IDSIA/USI-SUPSI, Switzerland, novembre 2003.

Alexander Pokahr, Lars Braubach et Winfried Lamersdorf. Jadex : A bdi reasoning engine. Dans Rafael Bordini, Mehdi Dastani, Jürgen Dix, Amal Fallah Seghrouchni et Gerhard Weiss, éditeurs, *Multi-Agent Programming*, volume 15 de *Multiagent Systems, Artificial Societies, and Simulated Organizations*, pages 149–174. Springer US, 2005. ISBN 978-0-387-26350-2.

Guillaume Pouyanne. *Forme Urbaine et Mobilité Quotidienne*. Thèse de doctorat, Université Montesquieu - Bordeaux IV, décembre 2004.

L.S.C. Pun-Cheng. A new face-entity concept for modeling urban morphology. *Journal of Urban and Regional Information Systems Association*, 12(3):47–56, 2000.

Anand S. Rao. Agentspeak(l) : Bdi agents speak out in a logical computable language. Dans *MAAMAW '96*, pages 42–55, Secaucus, NJ, USA, 1996. Springer-Verlag New York, Inc. ISBN 3-540-60852-4.

Seyed Naser Razavi, Nicolas Gaud, Abderrafiaa Koukam et Nasser Mozayani. Using motion levels of detail in the fast multipole method for simulation of large particle systems. Dans *the 15th World Multi-Conference on Systemics, Cybernetics and Informatics (WMSCI)*, Orlando, Florida, USA, jul 2011a. International Institute of Informatics and Cybernetics.

Seyed Naser Razavi, Nicolas Gaud, Nasser Mozayani et Abderrafiaa Koukam. Multi-agent based simulations using fast multipole method : Application to large scale simulations of flocking dynamical systems. *Artificial Intelligence Review*, 35(1):53–72, jan 2011b. ISSN 02692821.

Peter Reiher. Parallel simulation using the time warp operating system. Dans *1990 Winter Simulation Conference*, pages 38–45, New Orleans, USA, décembre 1990.

Craig W. Reynolds. Flocks, herds and schools : A distributed behavioral model. Dans *SIGGRAPH '87 : Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 25–34, New York, NY, USA, 1987. ACM. ISBN 0-89791-227-6.

Paul Reynolds. A spectrum of options for parallel simulation. Dans *1988 Winter Simulation Conference*, pages 325–332, San Diego, USA, décembre 1988.

Paul Reynolds et Philip Dickens. SRADS with local roolback. Dans *SCS Multi-Conference on Distributed Simulation*, pages 161–164, San Diego, USA, janvier 1990.

Luciene Cristina Rinaldi Rodrigues, Antonio Carlos Sementille et Ildeberto A.and Brega José Remo Ferreira Rodello. Managing large scale virtual environments using portals. Dans *VRCAI '04 : Proceedings of the 2004 ACM SIGGRAPH international conference on Virtual Reality continuum and its applications in industry*, pages 459–462, New York, NY, USA, 2004. ACM. ISBN 1-58113-884-9.

Sebastían Rodríguez, Nicolas Gaud, Vincent Hilaire, Stéphane Galland et Abderrafiaa Koukam. An analysis and design concept for self-organization in holonic multi-agent systems. Dans S. Bruckner, S. Hassas, M. Jelasity et D. Yamins, éditeurs, *Engineering Self-Organising Systems*, volume 4335 de *LNAI*, pages 15–27. Springer-Verlag, 2007.

Sebastían A. Rodríguez. *From analysis to design of Holonic Multi-Agent Systems : a Framework, methodological guidelines and applications*. Thèse de doctorat, Université de Technologie de Belfort-Montbéliard, 2005.

Stuart J. Russell et Peter Norvig. *Artificial intelligence - a modern approach : the intelligent agent book*. Prentice Hall series in artificial intelligence. Prentice Hall, 1995a. ISBN 978-0-13-103805-9.

Stuart J. Russell et Peter Norvig. *Artificial Intelligence : A Modern Approach*. Prentice Hall, (second edition 2003), 1st édition, January 1995b. ISBN 0137903952.

Behorkh Samadi. *Distributed Simulation, Performance and Analysis*. Thèse de doctorat, University of California, Los Angeles, USA, octobre 1985.

Hiroshi Sato et Akira Namatame. Finding micro-macro loop with market game. Dans *ICCIMA'01 : Proceedings of the Fourth International Conference on Computational Intelligence and Multimedia Applications*, page 236, Washington, DC, USA, 2001. IEEE Computer Society. ISBN 0-7695-1312-3.

R. Keith Sawyer. Artificial societies : Multiagent systems and the micro-macro link in sociological theory. *SOCIOLOGICAL METHODS AND RESEARCH*, 31:325–363, 2003.

A Schadschneider. Cellular automaton approach to pedestrian dynamics - theory. *Pedestrian Evacuation Dynamics*, (cond-mat/0112117):75. 11 p, Dec 2001.

T. Schelhorn, D. O'Sullivan, M. Haklay et M. Thurstain-Goodwin. Streets : An agent-based pedestrian model. Dans P. Rizzi, éditeur, *Proc. of the Conference on Computers in Urban Planning and Modelling*, 1999.

Michael Schillo, Klaus Fischer et ChristofT. Klein. The micro-macro link in dai and sociology. Dans Scott Moss et Paul Davidsson, éditeurs, *Multi-Agent-Based Simulation*, volume 1979 de *Lecture Notes in Computer Science*, pages 133–148. Springer Berlin Heidelberg, 2001. ISBN 978-3-540-41522-0.

Gero Schwenk. Micro-macro relations in the kirk-coleman model, 2004.

David Servat, Edith Perrier, Jean-Pierre Treuil et Alexis Drogoul. When agents emerge from agents : Introducing multi-scale viewpoints in multi-agent simulations. Dans *Proceedings of the First International Workshop on Multi-Agent Systems and Agent-Based Simulation*, pages 183–198, London, UK, UK, 1998. Springer-Verlag. ISBN 3-540-65476-3.

Joshua Shagam. *Dynamic spatial partitioning for real-time visibility determination*. Computer science, New Mexico State University, Department of computer science, avril 2003.

Robert E. Shannon. Simulation modeling and methodology. *SIGSIM Simul. Dig.*, 8(3):33–38, 1977. ISSN 0163-6103.

W. Shao et D. Terzopoulos. Environmental modeling for autonomous virtual pedestrians. Dans *SAE Symposium on Digital Human Modeling for Design and Engineering*, volume 114, pages 735–742. American Technical Publishers Ltd, 2005.

Wei Shao et Demetri Terzopoulos. Autonomous pedestrians. *Graph. Models*, 69(5-6):246–274, septembre 2007. ISSN 1524-0703.

Sharad Sharma et Stephen Otunba. Collaborative virtual environment to study aircraft evacuation for training and education. Dans Waleed W. Smari et Geoffrey Charles Fox, éditeurs, *CTS*, pages 569–574. IEEE, 2012. ISBN 978-1-4673-1381-0.

Onn Shehory et Sarit Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101(1–2):165–200, 1998. ISSN 0004-3702.

Nirajan Shiwakoti, Majid Sarvi, Geoff Rose et Martin Burd. Animal dynamics based approach for modeling pedestrian crowd egress under panic conditions. *Transportation Research Part B : Methodological*, 45(9): 1433 – 1449, 2011. ISSN 0191-2615.

Yoav Shoham. Agent oriented programming : An overview of the framework and summary of recent research. Dans Michael Masuch et László Pólos, éditeurs, *Knowledge Representation and Reasoning Under Uncertainty*, volume 808 de *Lecture Notes in Computer Science*, pages 123–129. Springer Berlin Heidelberg, 1994. ISBN 978-3-540-58095-9.

B. Singh. Interconnected Roles (ir) : A Coordination Model. Rapport technique CT-084-92, MCC, July 1992.

L. Sokol, D.P. Briscoe et A.P. Wieland. MTW : Strategy for scheduling discrete event simulation events for concurrent execution. Dans *SCS Multi-Conference on Distributed Simulation*, pages 34–43, San Diego, USA, février 1988.

Ian Sommerville. *Software Engineering*. International Computer Science Series. Addison Wesley, Pearson Education, seventh édition, 2004. ISBN 0-321-21026-3.

G. Still. *Crowd Dynamics*. Thèse de doctorat, Warwick University, 2000.

Olivier Szymanezyk, Patrick Dickinson et Tom Duckett. Towards agent-based crowd simulation in airports using games technology. Dans James O'Shea, Ngoc Thanh Nguyen, Keeley A. Crockett, Robert J. Howlett et Lakhmi C. Jain, éditeurs, *KES-AMSTA*, volume 6682 de *Lecture Notes in Computer Science*, pages 524–533. Springer, 2011. ISBN 978-3-642-21999-3.

Milind Tambe. Towards flexible teamwork. *J. Artif. Intell. Res. (JAIR)*, 7:83–124, 1997.

Franco Tecchia, Céline Loscos, Ruth Conroy et Yiorgos Chrysanthou. Agent behaviour simulator (abs) : A platform for urban behaviour development. Dans *In GTEC'2001*, pages 17–21, 2001.

Daniel Thalman, Nathalie Farenc et Ronan Boulic. Virtual human life simulation and database : Why and how. Dans *DANTE '99 : Proceedings of the 1999 International Symposium on Database Applications in Non-Traditional Environments*, page 471, Washington, DC, USA, 1999. IEEE Computer Society. ISBN 0-7695-0496-5.

Daniel Thalmann et Soraia Raupp Musse. *Crowd Simulation*. Springer, 2007. ISBN 978-1-84628-824-1.

Gwenola Thomas. *Environnements virtuels urbains : modélisation des informations nécessaires à la simulation de piétons*. Thèse de doctorat, Informatique, Université de Rennes 1, 16 décembre 1999.

Gwenola Thomas et Stéphane Donikian. Virtual humans animation in informed urban environments. Dans *CA '00 : Proceedings of the Computer Animation*, page 112, Washington, DC, USA, 2000. IEEE Computer Society.

P. Thompson et Marchant E. A computer-model for the evacuation of large building population. *Fire Safety Journal*, 24(2):131–148, 1995.

Sebastian Thrun et Arno Bü. Integrating grid-based and topological maps for mobile robot navigation. Dans *Proceedings of the thirteenth national conference on Artificial intelligence - Volume 2*, AAAI'96, pages 944–950. AAAI Press, 1996. ISBN 0-262-51091-X.

Harry Timmermans, Xavier van der Hagen et Aloys Borgers. Transportation systems, retail environments and pedestrian trip chaining behaviour : Modelling issues and applications. *Transportation Research Part B : Methodological*, 26(1):45 – 59, 1992. ISSN 0191-2615.

K.G. Troitzsch. Multilevel simulation. Dans K.G. Troitzsch, U. Mueller, N. Gilbert et J.E. Doran, éditeurs, *Social Science Microsimulation*. Springer-Verlag, Berlin, 1996.

Alasdair Turner et Alan Penn. Encoding natural movement as an agent-based system : An investigation into human pedestrian behaviour in the built environment. *Environment and Planning B : Planning and Design*, 29(4):473–490, 2002.

Adelinde M. Uhrmacher, Daniela Degenring et Bernard Zeigler. Discrete event multi-level models for systems biology. Dans C. Priami et al., éditeur, *Transactions on Computational Systems Biology I*, volume 3380 de *LNCS*, pages 66–89. Springer-Verlag Berlin Heidelberg, 2005.

Adelinde M. Uhrmacher et Corrado Priami. Discrete event systems specification in systems biology - a discussion of stochastic pi calculus and devs. Dans *WSC'05 : Proceedings of the 37th conference on Winter simulation*, pages 317–326. Winter Simulation Conference, 2005. ISBN 0-7803-9519-0.

M. Ulieru et A. Geras. Emergent holarchies for e-health applications : a case in glaucoma diagnosis. Dans *IEEE IECON'02*, volume 4, pages 2957–2961, 2002.

F. Van Aeken. *Les systèmes multi-agents minimaux*. Thèse de doctorat, Leibniz / IMAG, Institut National Polytechnique de Grenoble - INPG., 1999.

H. Van Dyke Parunak, Robert Savit et Rick L. Riolo. Agent-based modeling vs. equation-based modeling : A case study and users' guide. Dans J.S. Sichman, R. Conte et N. Gilbert, éditeurs, *Multi-Agent Systems and Agent-Based Simulation (MABS)*, pages 10–26, Paris, France, 1998. Springer Verlag.

Diane Vanbergue. *Conception de simulation multi-agents : application à la simulation des migrations intra-urbaines de la ville de Bogota*. Thèse de doctorat, Université Paris VI, December 2003.

Thomas Wagner et Victor R. Lesser. Relating quantified motivations for organizationally situated agents. Dans *6th International Workshop on Intelligent Agents VI, Agent Theories, Architectures, and Languages (ATAL),*, pages 334–348, London, UK, 2000. Springer-Verlag. ISBN 3-540-67200-1.

Fang Wang, Stephen Turner et Lihua Wang. Agent communication in distributed simulations. Dans Paul Davidsson, Brian Logan et Keiki Takadama, éditeurs, *Multi-Agent and Multi-Agent-Based Simulation*, volume 3415 de *Lecture Notes in Computer Science*, pages 11–24. Springer Berlin / Heidelberg, 2005. ISBN 978-3-540-25262-7.

Gerhard Weiss. *Multiagent Systems : A Modern Approach to Distributed Artificial Intelligence*, volume 3. MIT Press, 1999.

Michael P. Wellman. Online marketplaces. Dans Munindar P. Singh, éditeur, *Practical Handbook of Internet Computing*. Chapman Hall & CRC Press, Baton Rouge, 2004.

Michael P. Wellman et Peter R. Wurman. Market-aware agents for a multiagent world. *Robotics and Autonomous Systems*, 24:115–125, 1998.

D. Weyns, A. Omicini et J. Odell. Environment as a first-class abstraction in multiagent systems. *Journal on Autonomous Agents and Multiagent Systems*, 14(1), 2007.

Danny Weyns et Tom Holvoet. Formal model for situated multi-agent systems. *Formal Approaches for Multi-agent Systems, Special Issue of Fundamenta Informaticae*, 63(2-3), 2004a. Eds. B. Dunin-Keplicz, R. Verbrugge.

Danny Weyns et Tom Holvoet. A formal model for situated multi-agent systems. *Fundamenta Informaticae*, 63:1–34, 2004b.

Danny Weyns, H. Van Dyke Parunak, Fabien Michel, Tom Holvoet et Jacques Ferber. Environment for multiagent systems state-of-the-art and research challenges. Dans Springer Berlin / Heidelberg, éditeur, *Environments for Multi-Agent Systems (E4MAS)*, pages 1–47, 2005.

Danny Weyns, Elke Steegmans et Tom Holvoet. Towards active perception in situated multi-agent systems. *EUMAS, Special Issue of Journal on Applied Artificial Intelligence (AAI)*, 18(9-10):867–883, October–December 2004.

Gio Wiederhold, Peter Wegner et Stefano Ceri. Toward megaprogramming. *Commun. ACM*, 35:89–99, November 1992. ISSN 0001-0782.

U. Wilensky. Netlogo flocking model, 1998. URL `http://ccl.northwestern.edu/netlogo/models/Flocking`.

D. L. Wilson. Use of modeling and simulation to support airport security. Dans *International Carnahan Conference on Security Technology*, 2004.

M. Wooldridge et N. R. Jennings. Software engineering with agents : Pitfalls and pratfalls. *IEEE Internet Computing*, 3(3):20–27, 1999.

Jianguo Wu. Hierarchy and scaling : Extrapolating information along a scaling ladder. *Canadian Journal of Remote Sensing*, 25(4):367–380, 1999.

O. Yadgar, S. Kraus et C. Ortiz. *Scaling up distributed sensor networks : cooperative large-scale mobile-agent organizations*, pages 185–218. Kluwer publisching, 2003.

W J Yu, R Chen, L Y Dong et S Q Dai. Centrifugal force model for pedestrian dynamics. *Phys Rev E Stat Nonlin Soft Matter Phys*, 72(2 Pt 2):026112, 2005. ISSN 1539-3755.

Bernard P. Zeigler, Tag Gon Kim et Herbert Praehofer. *Theory of Modeling and Simulation*. Academic Press, 2nd edition édition, 2000.

L. Zhang, C.A. Athale et T.S. Deisboeck. Development of a three-dimensional multiscale agent-based tumor model : Simulating gene-protein interaction profiles, cell phenotypes and multicellular patterns in brain cancer. 244(1):96–107, 01 2007.

Le Zhang, Zhihui Wang, Jonathan A. Sagotsky et Thomas S. Deisboeck. Multiscale agent-based cancer modeling. *Journal of Mathematical Biology*, 58:545–559, 2009. ISSN 0303-6812.

Athanasios Ziliaskopoulos, Jiang Zhang et Huajing Shi. Hybrid mesoscopic-microscopic traffic simulation model : Design, implementation, and computational analysis. Dans *Transportation Research Board 85th Annual Meeting*, 2006.

TABLE DES FIGURES

LISTE DES TABLES

V

ANNEXES

A

DIAGRAMMES ASPECS DU MODÈLE DE L'ENVIRONNEMENT

Dans ce chapitre d'annexe, nous détaillons l'ensemble des schémas produits lors du processus de conception d'ASPECS. Les deux premières phases sont respectivement présentées dans les deux sections qui suivent. Les schémas produits lors de la dernière phase sont donnés dans le chapitre 6 de la thèse. Notons également que toutes les activités ne sont pas présentées. En effet, certaines activités sont parfois optionnelles ou détaillé dans le cœur de la thèse sous format textuel.

A.1/ PHASE D'ANALYSE DES BESOINS

Cette phase regroupe l'ensemble des schémas propres à l'analyse du problème. Ici, notre problème est la simulation multiniveau pour le déplacement de piétons au sein d'un environnement intérieur. La simulation multiniveau se focalise sur la représentation de l'environnement et ses dynamiques internes.

A.1.1/ ÉTAPE : DESCRIPTION DES BESOINS DU DOMAINE

Cette étape contient la description d'un diagramme de cas d'utilisation (*cf.* figure A.1).

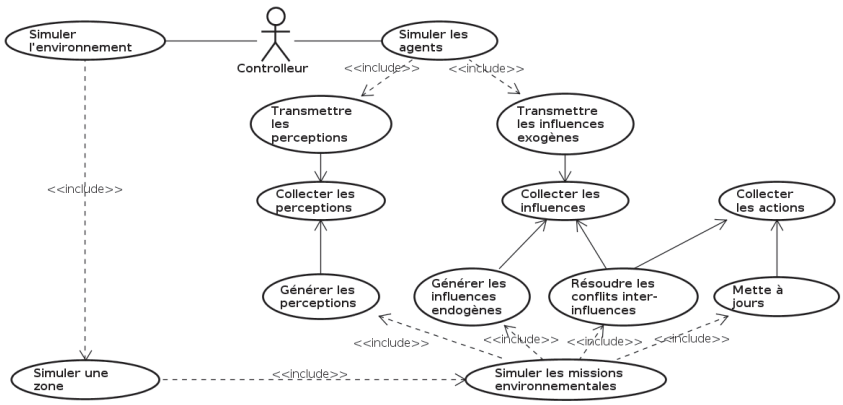


FIGURE A.1 – Descriptif des exigences du domaine sous forme d'un diagramme de cas d'utilisation.

Le contrôleur est une interface permettant à l'usager de lancer, mettre en pause ou arrêter la simulation. Il sert également à ordonnancer la simulation des piétons et la dynamique interne de l'environnement.

A.1.2/ ÉTAPE : DESCRIPTION DE L'ONTOLOGIE

Les figures A.2, A.3 et A.4 décrivent l'ontologie de notre modèle de simulation. L'ontologie présentée dans cette section se substitue aux concepts de CRIO. Le lien entre CRIO et le domaine du problème, en partie défini par l'ontologie, est présenté dans le chapitre 4.

La figure A.2 décrit l'ontologie du problème sans rentrer dans les détails sur les missions environnementales, ni même la décomposition hiérarchique.

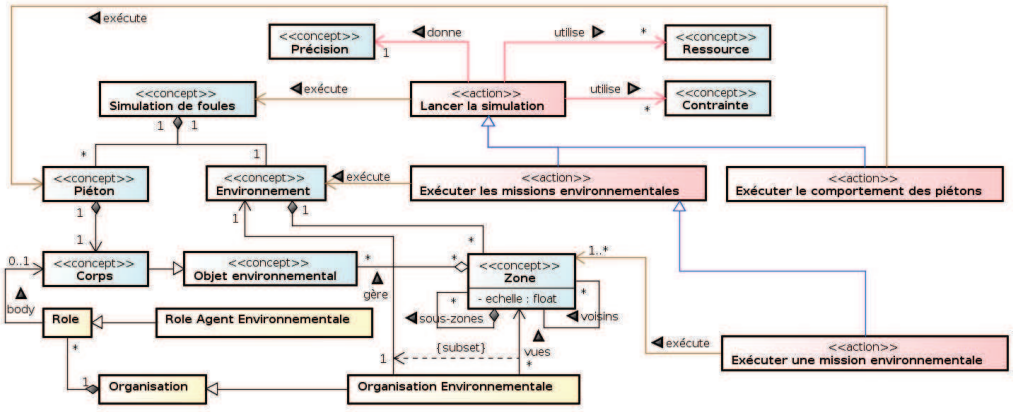


FIGURE A.2 – Ontologie générale pour la simulation de foules.

La figure A.3 détaille l'ontologie sur les missions environnementales.

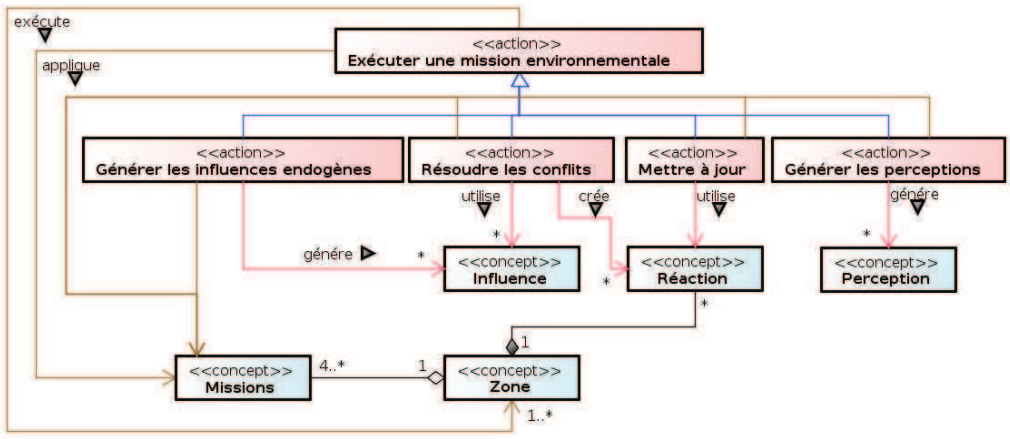


FIGURE A.3 – Ontologie détaillée pour l'exécution des missions environnementales.

La figure A.4 propose une décomposition de l'environnement. Concernant la décompo-

sition des zones (*e.g.* Département, Arrondissement, *etc.*), nous nous somme basé sur la décomposition administrative française. Les objets environnementaux sont peu nombreux, nous les avons limité aux corps des piétons, aux groupes de corps et aux regroupements d'entités. À titre indicatif, nous avons également ajouté le mobilier, la signalétique routière et la végétation.

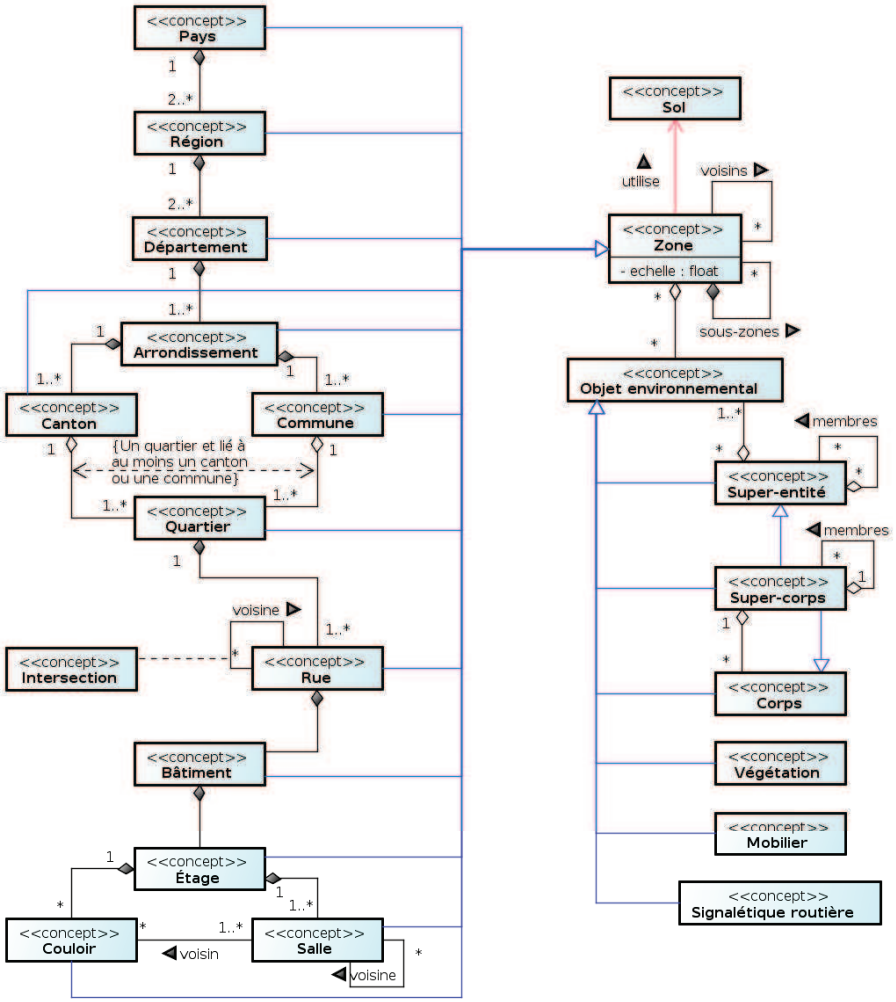


FIGURE A.4 – Ontologie détaillée pour la décomposition de l'environnement.

A.1.3/ ÉTAPE : IDENTIFICATION DES INTERACTIONS ET DES RÔLES

D'après le diagramme de cas d'utilisation (*cf.* figure A.1), des regroupements structurels et fonctionnels sont effectués pour définir les organisations de notre modèle organisationnel. Ces organisations sont présentées dans la figure A.5.

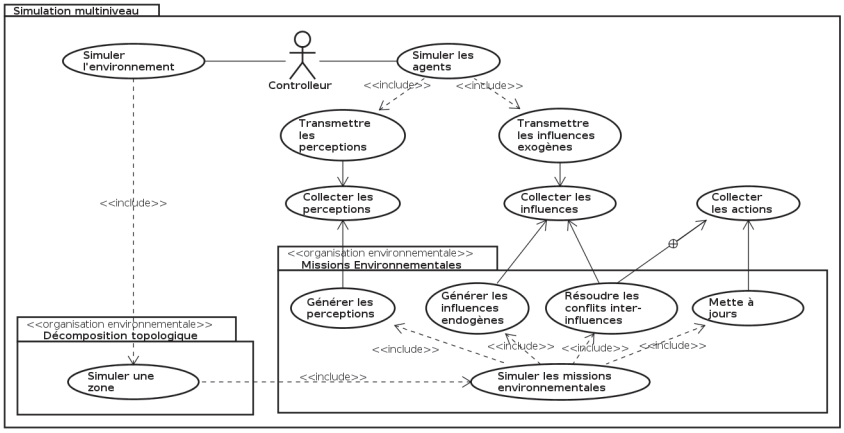


FIGURE A.5 – Identification des organisations de l'environnement pour une simulation de foules.

A.1.4/ ÉTAPE : DESCRIPTION DES SCÉNARIOS D'INTERACTIONS

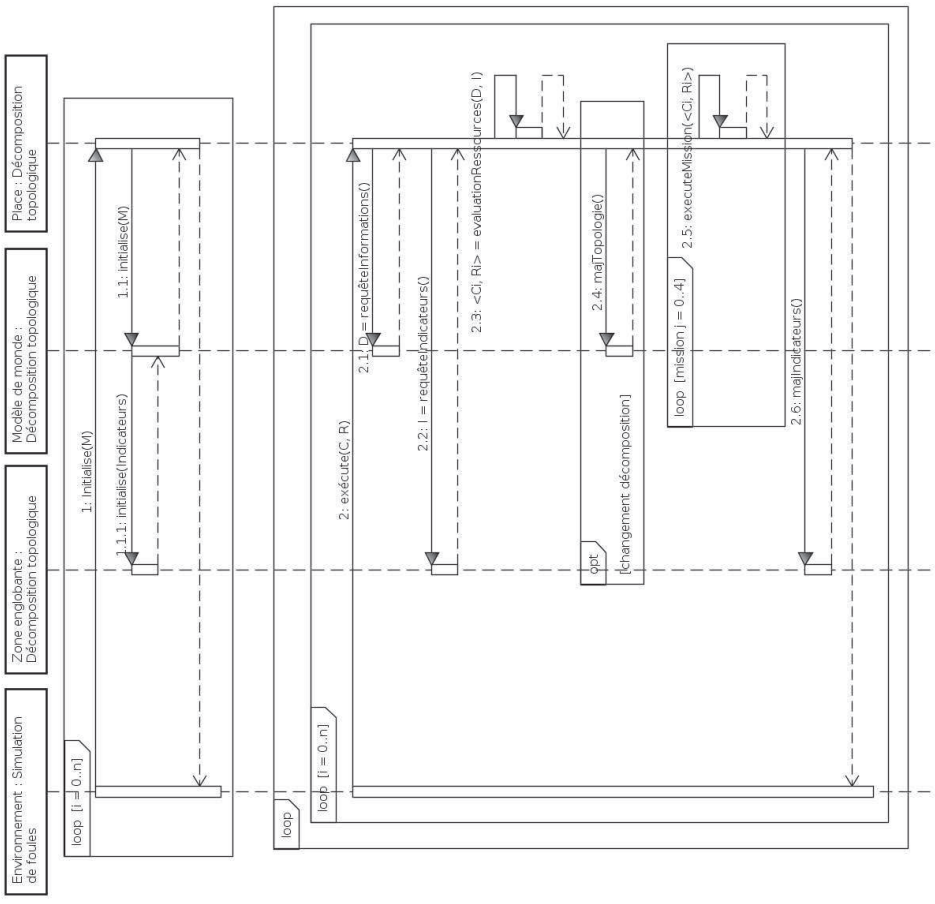


FIGURE A.6 – Description des interactions au sein de l'organisation *décomposition topologique*

Les figures A.7, A.6 et A.8 détaillent les scénarios d'interactions des organisations du modèle d'environnement.

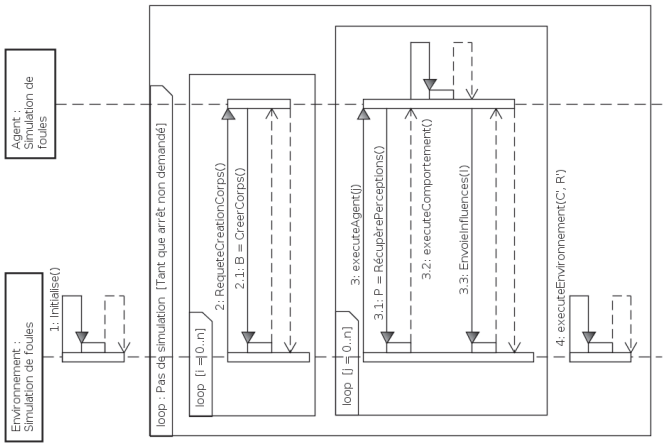


FIGURE A.7 – Description des interactions au sein de l'organisation *Simulation multiniveau*

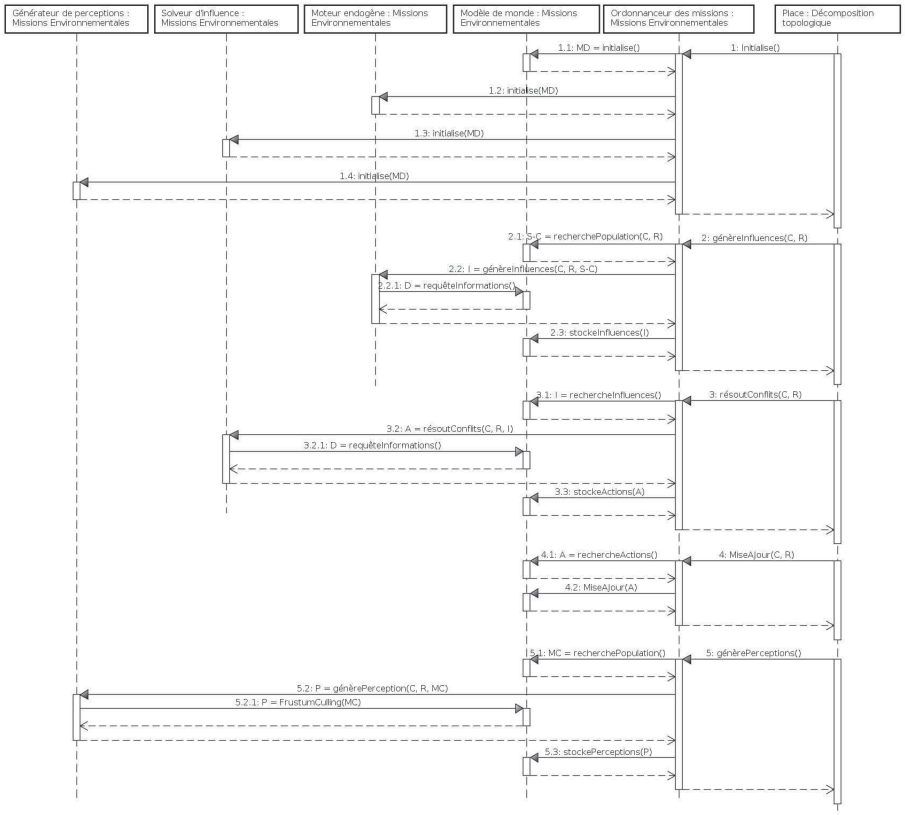


FIGURE A.8 – Description des interactions au sein de l'organisation *Missions environnementales*

A.1.5/ ÉTAPE : DESCRIPTION DES PLANS DE COMPORTEMENT DES RÔLES

Cette section décrit les plans de comportement entre rôles d'une même organisation (*cf.* figures A.9 et A.10).

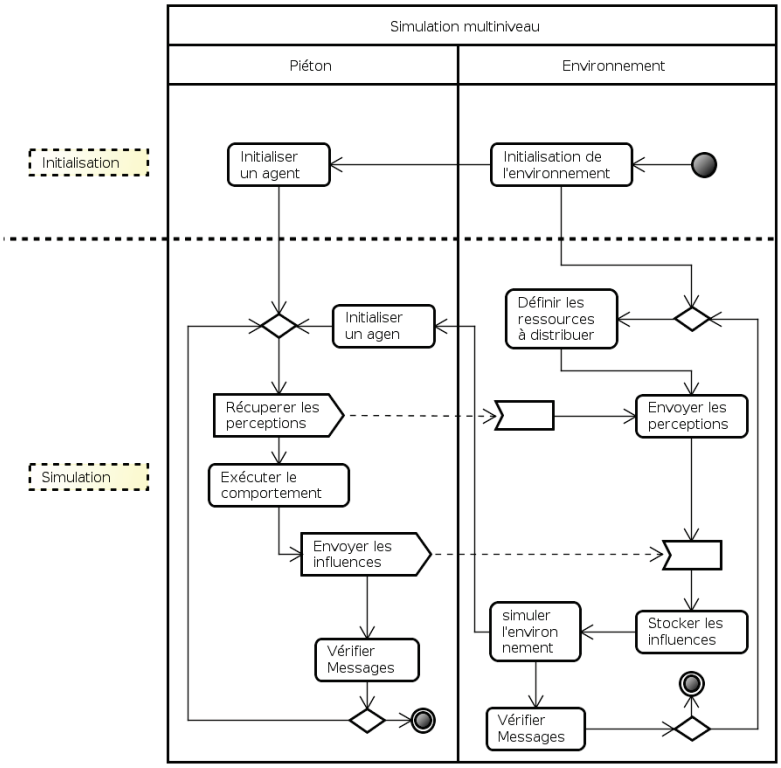


FIGURE A.9 – Plan de comportement des rôles de l'organisation *simulation multiniveau*.

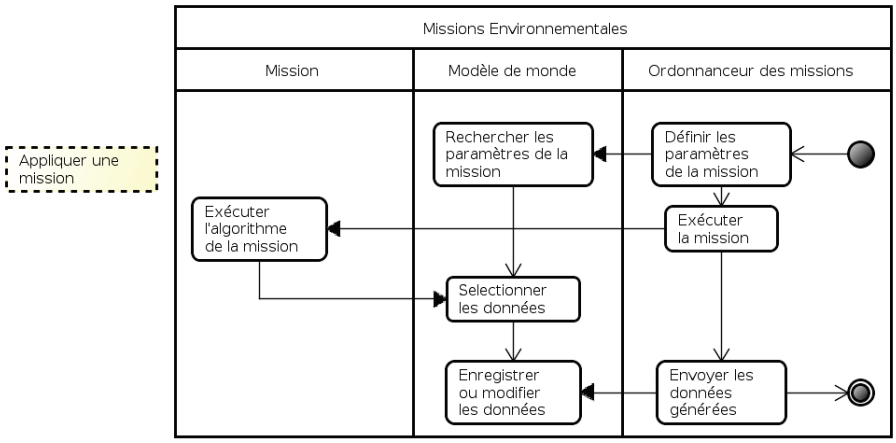


FIGURE A.10 – Plan de comportement partiel d'un rôle de l'organisation *missions environnementales*, pour l'application d'une mission environnementale.

La figure A.10 représente une généralisation du comportement des rôles : *générateur de perceptions*, *moteur endogène* et *solveur d'influences*. Ces rôles sont représentés par le rôle *mission*. Les paramètres, les données et l'algorithme sont dépendants de la mission effectuée. Les flèches pleines indiquent une synchronisation entre les actions. Ces figures complètent celles produites lors de l'étape précédente.

A.1.6/ ÉTAPE : IDENTIFICATION DES CAPACITÉS

Nom : `Mettre à jour l'état du monde`

Paramètres :

- ME : un ensemble de super-entités
- A : les actions à appliquer
- M : le monde contenant les super-entités à un instant t

Résultats :
$M_{t+1} = M_t \oplus A$
A est l'ensemble des actions produites par l'environnement

Requiert : Les actions sont associées à des super-entités situées dans le modèle de monde M

Garantit : Les topologies de M_{t+1} et M_t sont identiques.

Description Textuelle :
Modifie l'état du monde suivant les actions à appliquer sur les super-entités.

FIGURE A.11 – La capacité `Appliquer les actions`

Nom : `Mettre à jour les structures du monde`

Paramètres :

- N : un niveau
- M_t : le monde contenant les super-entités à un instant t

Résultats :
$M_{t+1} = M_t \oplus N$

Requiert : Les actions sont associées à des super-entités situées dans le modèle de monde M'

Garantit : Les entités situés dans M_t et M_{t+1} sont identiques (position, taille, attributs, *etc.*).

Description Textuelle :
Modifie l'état du monde suivant les niveaux à modifier (les super-entités sont également modifiés).

FIGURE A.12 – La capacité `mettre à jour la topologie`

Nom : `Générer les influences endogènes`

Paramètres :

- ME : un ensemble de super-entités
- M : le monde contenant les super-entités
- L : les lois du monde M

Résultats :

$I = \{I_0 = L_1(ME_1), ..., I_n = L_1(ME_n), ..., I_{m.(n-1)+1} = L_m(ME_1), ..., I_{m.n} = L_m(ME_n)\}$

, avec $\forall k \in \{0..n\}, ME_k \in ME$ et $\forall j \in \{0..m\}, L_j \in L$

I est l'ensemble des influences endogènes produites par les super-entités ME

Requiert : L'ensemble des super-entités est contenu dans le monde M.

Garantit : *RAS*

Description Textuelle :
Génère l'ensemble des influences produites par l'application des lois du monde sur les super-entités.

FIGURE A.13 – La capacité `Générer les influences endogènes`

Nom : `Rechercher des super-entités`

Paramètres :

- ME : un ensemble de super-entités
- M : le monde contenant les super-entités
- G : une forme géométrique convexe

Résultats :

$z(M) \bigcup G = ME$

Requiert : Une fonction $z : M- > G$ prenant en paramètre un monde M et retournant la forme géométrique G correspondant recouvrant la zone spatiale à laquelle est associée le monde M.

Garantit : Les super-entités sont contenues dans le monde M et situées dans sa zone spatiale associée Z.

Description Textuelle :
Recherche les super-entités en intersection avec la zone spatiale associée à un monde et une forme géométrique convexe.

FIGURE A.14 – La capacité `Rechercher des super-entités`

A.2/ PHASE DE CONCEPTION DE LA SOCIÉTÉ AGENT

Cette phase regroupe l'ensemble des schémas propres à la création du modèle holonique suivant l'analyse du problème effectuée dans la section précédente.

A.2.1/ ÉTAPE : DESCRIPTION ONTOLOGIQUE DES COMMUNICATIONS

Le schéma détaillant les communications (*cf.* figure A.15) réutilise celui donné lors de la description des organisations.

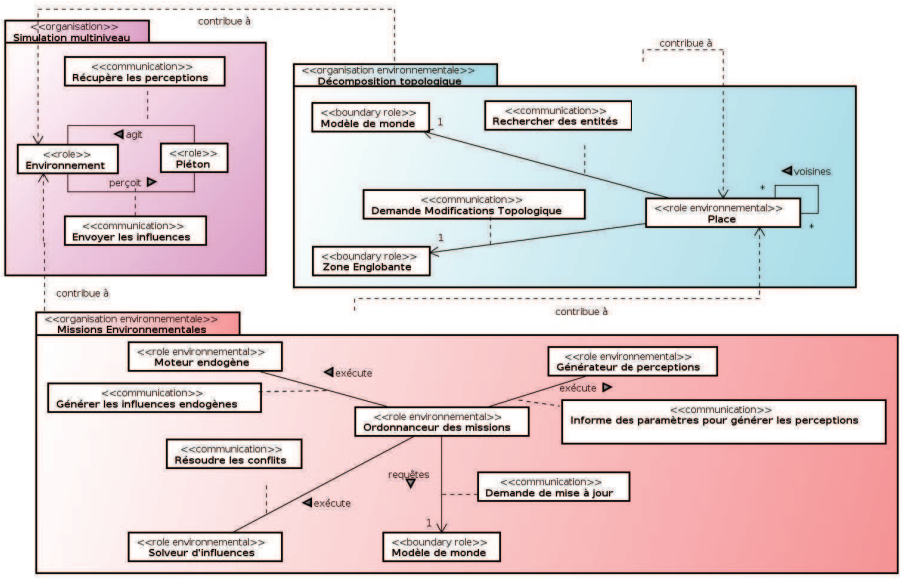


FIGURE A.15 – Description des communications des organisations du modèle de simulation

A.2.2/ ÉTAPE : DESCRIPTION DES COMPORTEMENTS DES RÔLES

Cette étape donne le comportement spécifique à chaque rôle. Le comportement des rôles *moteur endogène* et *solveur d'influences* sont une analogie à celui du *générateur de perception*.

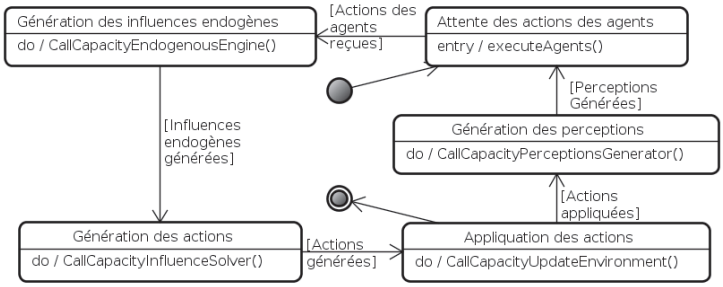


FIGURE A.16 – Description du comportement du rôle *environnement*

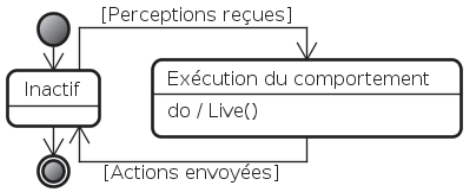


FIGURE A.17 – Description du comportement du rôle *agent*

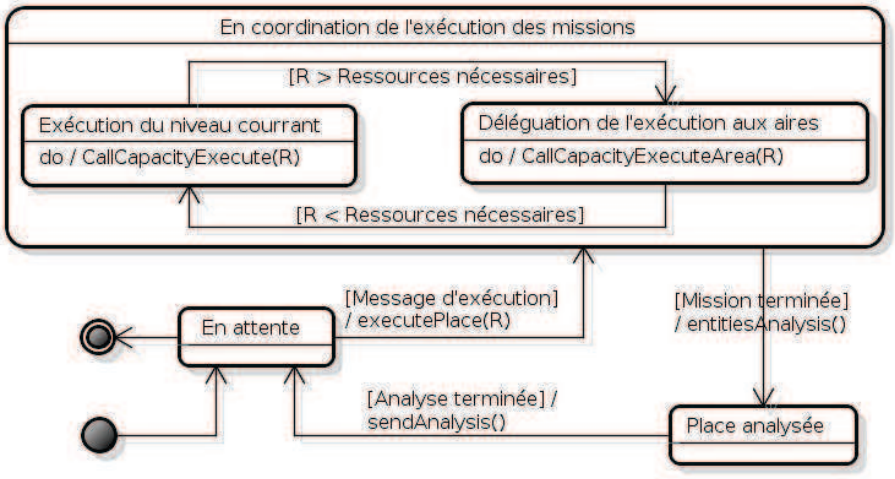


FIGURE A.18 – Description du comportement du rôle *place*

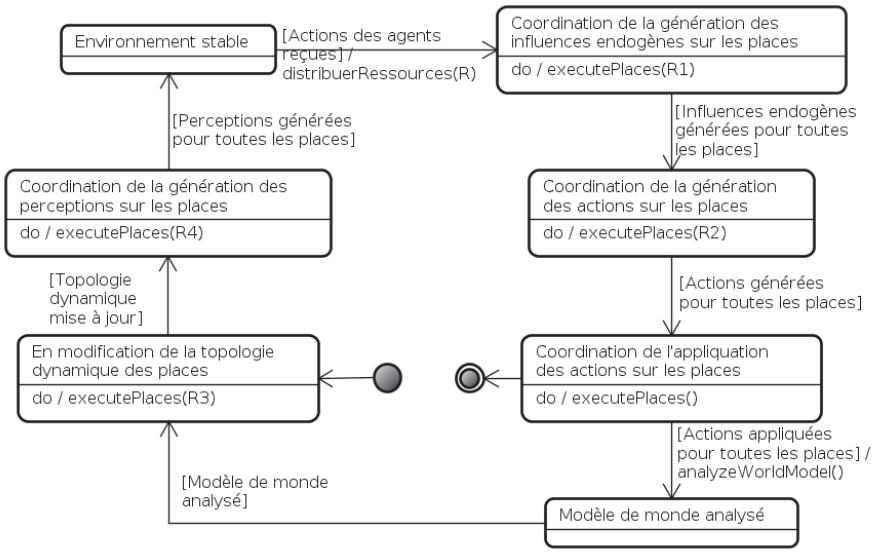


FIGURE A.19 – Description du comportement du rôle *zone englobante*

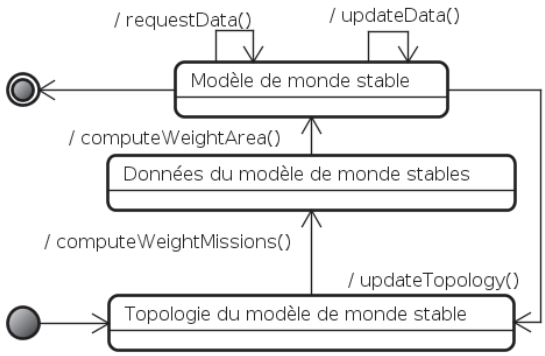


FIGURE A.20 – Description du comportement du rôle *modèle de monde*

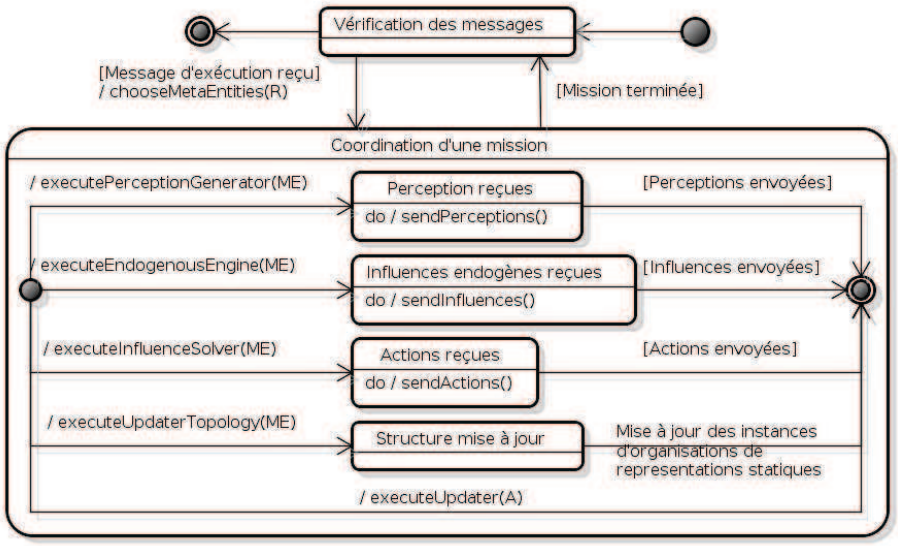


FIGURE A.21 – Description du comportement du rôle *ordonnanceur des missions*

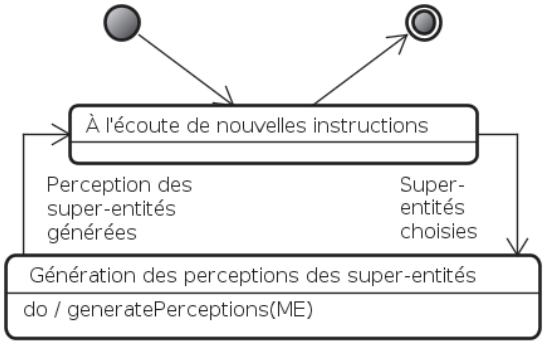


FIGURE A.22 – Description du comportement du rôle *générateur de perceptions*

B

ALGORITHMES ASSOCIÉS AUX MISSIONS DE L'ENVIRONNEMENT

Cette annexe décrit l'ensemble des algorithmes implantés dans la plate-forme JASIM. La figure B.1 illustre l'architecture générale de la plate-forme. L'ensemble des algorithmes présentés dans cette annexe ont été implantés dans le module *« Object-Oriented Environment Model »* .

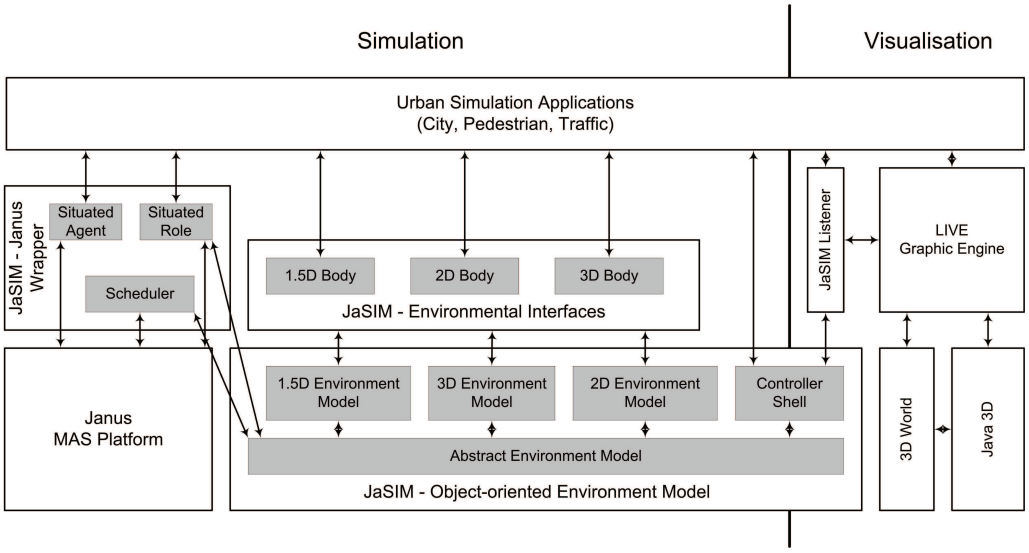


FIGURE B.1 – Architecture globale de la plate-forme JASIM permettant de simuler un SMA dans un environnement virtuel [Galland et al., 2009]

Cette annexe est structurée comme suit. Tout d'abord, les structures de données et leurs algorithmes sont présentés. Ensuite, nous présentons les algorithmes qui permettent de calculer les perceptions des agents à partir du contenu des structures de donnnées. L'avant-dernière section présente les algorithmes permettant de détecter et résoudre les problèmes de cohérence causés par les actions des agents. La dernière section de cette annexe est consacrée aux algorithmes de mise-à-jour des structures arborescentes de données.

B.1/ STRUCTURE DE DONNÉES DÉCRIVANT L'ENVIRONNEMENT

La plateforme JASIM propose un modèle de données permettant de stocker l'ensemble des objets présents dans l'environnement.

B.1.1/ OBJETS DANS L'ENVIRONNEMENT

Chaque objet dans l'environnement possède une position et une forme géométrique représentant son volume : sa boîte englobante.

```
class WorldEntity
begin
  // position of the object
  position : Point

  // bounding box that is enclosing the object (commonly AABB)
  box : BoundingBox
end
```

Certains objets ont la capacité de se déplacer ou d'être déplacés : les `MobileEntity`. Ces objets possède une orientation et un vecteur de vélocité. Ces deux informations sont distinguées afin de permettre à l'objet de s'orienter indépendamment de sa direction de mouvement. Nous ajoutons une représentant simplifiée de son maillage 3D (sa forme géométrique composée de triangles) afin de permettre une mise-à-jour précise de sa boîte englobante après une rotation. En effet, la rotation appliquée à un objet ne peut pas être directement appliquée à la boîtes englobantes si elle est alignée sur les axes du mondes. Si la rotation était directement appliquée, la boîte englobante approximerait plus grossièrement encore la volume de l'objet, et la conséquence serait qu'à chaque rotation, le volume occupé par la boîte augmenterais, sans aucune possibilité de diminution. Le maillage permet de recalculer la boîte englobante au plus juste après chaque rotation.

```
class MobileEntity extends WorldEntity
begin
  // orientation of the environment object
  orientation : Vector

  // velocity vector of the entity
  velocity : Vector

  // simplified mesh which permits to re-compute the bounding
  // box when the entity is rotating (when using AABB).
  mesh : Array of Point
end
```

Le corps d'un agent est un objet mobile particulier. Il est contrôler par un agent et fournit à ce dernier la possibilité de percevoir et d'agir dans le monde.

```
class AgentBody extends MobileEntity
begin
  // Field of perception
  frustum : Frustum

  // Set of the lastly computed perceptions
```

```
  perceptions : Array of WorldEntity

  // Set of influences given by the agent
  influences : Map from MobileEntity to Vector
end
```

B.1.2/ MODÈLE MATRICIELLE DE L'ENVIRONNEMENT

La structure de données la plus simple pour stocker et localiser les objets de l'environnement est une matrice. Chaque cellule de la matrice représente une zone, géométriquement identique aux autres, de l'environnement. Chaque cellule contient un ensemble d'objets et quelques attributs (type de sol, *etc*).

```
class Cell
begin
  // A cell is able to contains a set of objects.
  // Basically a cell contains zero or one object. But in several
  // applications the set could be larger.
  objects : Array of WorldEntity

  // Attributes associated to the ground enclosed by the cell
  groundDescription : Map from String to Object
end
```

```
class Grid
begin
  // Sizes of the world
  width : Positive Number
  height : Positive Number

  // Set of cells
  cells : matrix of Cell [width][height]
end
```

Cette structure matricielle est très performante. Mais elle souffre de deux problèmes majeures : elle discrétise l'espace, et elle occupe un espace en mémoire très important.

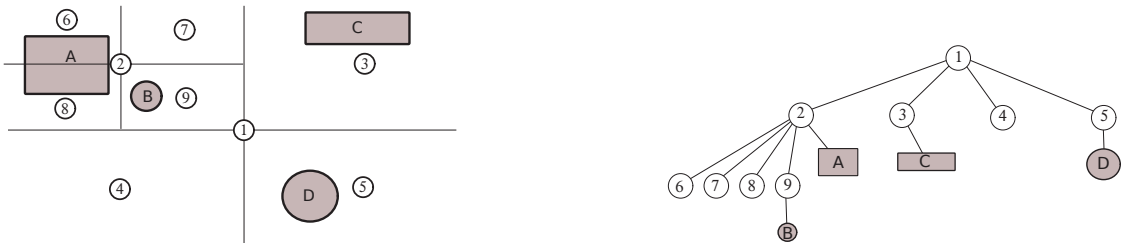
B.1.3/ MODÈLE ARBORESCENT DE L'ENVIRONNEMENT

La seconde structure de données classiquement utilisée est l'arbre spatial. L'arbre spatial structure l'espace en le découpant en sous-zones. Chaque nœud de l'arbre représente une zone et ses nœuds fils les sous-zones issues de la décomposition. La figure B.2 illustre la décomposition d'un environnement par un arbre spatial divisant l'espace en quatre parties de même dimension.

B.1.3.1/ STRUCTURE D'ARBRE SPATIAL

Un arbre spatial contient un nœud racine.

```
class Tree
begin
```



(a) Vue du dessus de l'environnement (b) Structure de l'arbre

FIGURE B.2 – Exemple de décomposition dynamique d'une zone par un arbre kD-tree, avec $k = 4$

```
  // A tree contains a set of nodes accessible through the root node
  rootNode : TreeNode
end
```

Chaque nœud de l'arbre couvre une zone de l'environnement, représentée par la boîte englobante associée au nœud. Enfin, chaque nœud contient un ensemble, potentiellement vide, d'objets de l'environnement se trouvant dans la zone occupée par le nœud. Ici, nous ne précisons pas l'algorithme permettant de construire un arbre spatial. Ces algorithmes sont dépendants de l'heuristique permettant de découper l'espace. Ils sont présentés dans la suite de cette annexe.

```
class TreeNode
begin
  // A node could contains a set of children
  children : List of TreeNode

  // A node has a parent
  parent : TreeNode

  // The tree node is supposed to enclose a part of the environment space.
  // This is described by a bounding box. The bounding box of
  // a tree node must enclose all the bounding boxes of its children.
  box : BoudingBox

  // Several nodes contains objects so that the bounding box of the node is
  // enclosing all the bounding boxes of the objects
  objects : Set of WorldEntity

end
```

B.1.3.2/ ITÉRATION SUR LES NŒUDS D'UN ARBRE SPATIAL

L'un des « *Design Pattern* » les plus utilisés est celui de l'itération. Un itérateur est un objet permettant de parcourir une structure de donnée indépendamment de la manière dont elle a été codée.

L'interface ci-dessous définit un itérateur. Nous nous sommes inspiré de l'implantation proposée dans les spécifications du language Java.

```
interface Iterator
begin
```

```
  // Replies if there is one more element to iterate on
  function hasNext() : boolean
  // Replies the next available element
  function next() : TreeNode
end
```

Ci-dessous, nous présentons un exemple de l'utilisation d'un itérateur sur un arbre spatial. Un grand avantage de ce « *Design Pattern* » est de rendre l'algorithme parcourant les éléments indépendant de l'implantation interne de la structure de données.

```
function myFunction(tree)
begin
  // Create an iterator
  Iterator iter = tree.iterator()

  while iter.hasNext()
  do
    node = iter.next()
    // Do something with the node
  done
end
```

L'itérateur étant défini comme une interface, il est nécessaire de fournir une implantation spécifique pour chaque mode de parcours d'un arbre : parcours en largeur d'abord, ou parcours en profondeur d'abord.

B.1.3.3/ PARCOURS EN LARGEUR D'ABORD DE L'ARBRE SPATIAL

L'algorithme ci-dessous permet de parcourir les nœuds d'un arbre en largeur d'abord : les nœuds sont retournés dans l'ordre et niveau après niveau.

```
class BroadFirstIteration implements Iterator
begin
  // tree on which the iteration must be applied.
  tree : Tree

  // QUEUE of nodes that are encountered during the iteration but not
  // yet treated
  availableNodes : Queue of TreeNode

  // Initialise the iterator
  function initialise()
  begin
    availableNodes += tree.root;
  end

  // Replies if a element is available in the iterator
  function hasNext()
  begin
    return ! availableNodes.isEmpty()
  end

  // Replies the next available element from this iterator
  function next()
  begin
```

```
    // Test if a node is available
    if (availableNodes.isEmpty()) return None;

    // Extract the next available node
    current = availableNodes.poll();

    // Add the child nodes as available nodes
    foreach child from current.getChildrenNodes()
      availableNodes.enqueue(child);
    end for

    return current;
  end
end
```

B.1.3.4/ PARCOURS EN PROFONDEUR D'ABORD DE L'ARBRE SPATIAL

L'algorithme ci-dessous permet de parcourir les nœuds d'un arbre en profondeur d'abord : les nœuds fils de n sont retournés dans l'ordre avant les nœuds frères de n.

```
class PrefixDepthFirstIterator implements Iterator
begin
  // root before children

  // tree on which the iteration must be applied.
  tree : Tree

  // STACK of nodes that are encountered during the iteration but not
  // yet treated
  availableNodes : Stack of TreeNode

  // Initialise the iterator
  function initialise()
  begin
    availableNodes += tree.root;
  end

  // Replies if a element is available in the iterator
  function hasNext()
  begin
    return ! availableNodes.isEmpty()
  end

  // Replies the next available element from this iterator
  function next()
  begin
    // Test if a node is available
    if (availableNodes.isEmpty()) return None;

    // Extract the next available node
    current = availableNodes.pop();

    // Add the child nodes as available nodes
    foreach child from current.getChildrenNodes()
```

```
      availableNodes.push(child);
    end for

    return current;
  end
end
```

B.1.4/ DÉFINITION D'UN ARBRE SPATIAL $2D$-TREE

Un arbre spatial $2D$-tree est un arbre découpant l'espace en deux parties d'égale dimension. La limite entre ces deux parties est représentée par un plan, définit par :

```
interface Plane
begin
  // Factors related to the equation of the plane in 3D: Ax+By+Cz+D=0
  A : Number
  B : Number
  C : Number
  D : Number
  function classifies(box) : {IN_FRONT | BEHIND | INTERSECTS}
end
```

Chaque plan est capable de déterminer si un volume (« *box* ») se trouve entière devant, derrière ou en intersection avec le plan. L'avant et l'arrière d'un plan est déterminé par la normale au plan. Ainsi l'avant du plan correspond au demi-espace positif, et l'arrière au demi-espace négatif par rapport à la normale.

Pour obtenir un arbre $2D$-tree, il suffit de spécialiser un nœud de l'arbre pour contenir son plan de découpe.

```
class BSPTreeNode extends TreeNode
begin
  // Cut plane, implementation depends on the size of the space (2D or 3D)
  cutPlane : Plane
end class
```

B.1.4.1/ ALGORITHME DE CONSTRUCTION D'UN $2D$-TREE

L'algorithme permettant de construire un arbre spatial $2D$-tree est :

```
class BSPTreeBuilder
begin
  // Build a tree that contains the given entities
  // splitCount is the maximal number of objects to store in the nodes
  function buildTree(entities, splitCount)
  begin
    // Create a node and initialize the cut planes
    tree = new Tree
    tree.root = new BSPTreeNode()
    tree.root.cutPlane = None
    tree.root.objects += entities

    // Split the node if necessary
```

```
  splitNode(tree.root, splitCount)

  // Return the new tree
  return tree
end

// Split a node if the count of entities is too big
function splitNode(node, splitCount)
begin
  // Does the number of elements in the node implies a split of this node
  if node.objects.length > splitCount

    // The number of objects in the node is too big, we
    // must split the node
    // Compute the best cut plane
    node.cutPlane = computeBestCutPlane(node.objects)

    // Go through all the entities to classify them against the cut plane
    notDispatchableObjects = [ ]
    for entity in node.objects
      switch(node.cutPlane.classify(entity.box))
      case IN_FRONT do
        if (node.children[0] = None)
          node.children[0] = new BSPTreeNode()
          node.children[0].cutPlane = None
        end if
        node.children[0].objects += entity

      case BEHIND do
        if (node.children[1] = None)
          node.children[1] = new BSPTreeNode()
          node.children[1].cutPlane = None
        end if
        node.children[1].objects += entity

      case INTERSECTS:
        notDispatchableObjects += entity

      end switch
    end for

    node.objects = notDispatchableObjects

    // Split the child nodes if necessary
    for child in node.children
      splitNode(child,splitCount)
    end for
end

// Compute a cut plane according to a given partition heuristic
function computeBestCutPlane(entities)
begin
  // Cut the space according to the biggest size of the AABB of
  // entities
  aabb = BoundingBox.union(entities)
```

```
    if (aabb.getSizeX() > aabb.getSizeY() and
      aabb.getSizeX() > aabb.getSizeZ())
      plane = new Plane(-1x + 0y + 0z - (aabb.lower.x + aabb.upper.x) / 2)

    else if (aabb.getSizeY() > aabb.getSizeX() and
      aabb.getSizeY() > aabb.getSizeZ())
      plane = new Plane(0x - 1y + 0z - (aabb.lower.y + aabb.upper.y) / 2)

    else
      plane = new Plane(0x + 0y - 1z - (aabb.lower.z + aabb.upper.z) / 2)

    return plane
  end
end
```

B.1.5/ DÉFINITION D'UN ARBRE SPATIAL $4D$-TREE

Un arbre spatial $4D$-tree est un arbre découpant l'espace en quatre parties d'égale dimension. Ici, nous n'utilisons pas deux plans pour représenter les limites de ces parties. Nous préférons utiliser la coordonnée du point d'intersection de ces plans. En effet, une des propriétés d'un $4D$-tree est d'utiliser des plan parallèles aux axes géométriques du monde. La connaissance des coordonnées du point d'intersection des deux plans de découpe est donc suffisante pour déterminer dans laquelle des quatre sous-parties un point est localisé. Nous assumons que l'heuristique de découpe est toujours dans le plan horizontal. La troisième coordonnée z devient inutile.

```
class QuadTreeNode extends TreeNode
  // Holds the coordinate of the plane that cut the plane along the X axis
  cutX : Number

  // Holds the coordinate of the plane that cut the plane along the Y axis
  cutY : Number
end
```

B.1.5.1/ ALGORITHME DE CONSTRUCTION D'UN $4D$-TREE

L'algorithme permettant de construire un arbre spatial $4D$-tree est :

```
class QuadTreeBuilder
begin
  // Holds the bounds of the world. It could be initialize with the union of the
  // entities' bounding boxes
  worldBounds : BoundingBox

  function buildTree(entities, splitCount)
  begin
    // Create a node and initialize the cut lines on the plane
    tree = new Tree
    tree.root = new QuadTreeNode()
    tree.root.cutX = worldBounds.getCenterX()
    tree.root.cutY = worldBounds.getCenterY()
    tree.root.objects += entities
```

```
  // Split the node if required
  splitNode(tree.root, splitCount)

  // Return the new tree
  return tree
end

// Replies the index of the child which is enclosing the given entity
function getEnclosingChildIndex(entity, node)
begin
  bounds = entity.box

  if (bounds.getMaxX() < node.cutX && bounds.getMaxY() < node.cutY)
    return [0] // SOUTH WEST
  if (bounds.getMinX() > node.cutX && bounds.getMaxY() < node.cutY)
    return [1] // SOUTH EAST
  if (bounds.getMaxX() < node.cutX && bounds.getMaxY() > node.cutY)
    return [2] // NORTH WEST
  if (bounds.getMinX() > node.cutX && bounds.getMaxY() > node.cutY)
    return [3] // NORTH EAST

  // The entity intersects at least one of the cut lines
  if (bounds.getMaxX() < node.cutX) return [0,2]
  if (bounds.getMinX() > node.cutX) return [1,3]
  if (bounds.getMaxY() < node.cutY) return [0,1]
  if (bounds.getMinY() > node.cutY) return [2,3]
  return [0,1,2,3]
end

function splitNode(node, splitCount)
begin
  // Does the number of elements in the node implies a split of the node
  if node.objects.length > splitCount

    // The number of objects in the node is too big, we
    // must split the node

    // Go through all the entities to classify them against the child
    // nodes
    for entity in node.objects

      // Get the indexes of the children that enclosing the entity
      for index in getEnclosedChildIndexes(entity,node)

        // Create the child node if never created before
        child = node.children[index]
        if child == None
          child = new QuadTreeNode
          node.children[index] = child
          child.cutX = computeCutX(node.cutX, index)
          child.cutY = computeCutY(node.cutY, index)
        end if

        // Insert the entity into the child node
        child.objects += entity
```

```
      end for

    end for

    // Split the child nodes if necessary
    for child in node.children
      splitNode(child,splitCount)
    end for

  end if
end

// Compute the coordinate of the cut lines for a child node
function computeCutX(x, childIndex)
begin
  if (childIndex==0 || childIndex==2)
    return (worldBounds.getMinX()+x) / 2
  else
    return (worldBounds.getMaxX()+x) / 2
end

// Compute the coordinate of the cut lines for a child node
function computeCutY(y, childIndex)
begin
  if (childIndex==0 || childIndex==2)
    return (worldBounds.getMinY()+y) / 2
  else
    return (worldBounds.getMaxY()+y) / 2
end

end
```

B.1.5.2/ Modification de l'algorithme de construction pour créer un $4D$-icoseptree

```
// Replies the index of the child which is enclosing the given entity
function getEnclosingChildIndex(entity, node)
begin
  bounds = entity.box

  if (bounds.getMaxX() < node.cutX && bounds.getMaxY() < node.cutY)
    return [0]
  if (bounds.getMinX() > node.cutX && bounds.getMaxY() < node.cutY)
    return [1]
  if (bounds.getMaxX() < node.cutX && bounds.getMaxY() > node.cutY)
    return [2]
  if (bounds.getMinX() > node.cutX && bounds.getMaxY() > node.cutY)
    return [3]

  // The entity intersects at least one of the cut lines
  return [4]
end
```

B.2/ CALCUL DES PERCEPTIONS

B.2.1/ DÉFINITION D'UN CHAMPS DE PERCEPTION

```
interface Frustum
begin
  function intersects(box : BoundingBox) : boolean
end
```

B.2.2/ DÉFINITION D'UN CHAMPS DE PERCEPTION PYRAMIDAL

```
class PyramidalFrustum implements Frustum
begin
  // Holds the planes, ie a set of 4-element vectors representing the
  // elements of the plane equation (A,B,C,D)
  near : Plane
  far : Plane
  left : Plane
  right : Plane
  top : Plane
  bottom : Plane

  // Compute the planes according to the view direction, the position of the
  // eyes and the field of view
  function createFrustum(eye, viewDirection, openessAngle, nearDistance,
       farDistance)
  begin
    (near.a, near.b, near.c) =
      (viewDirection.x,viewDirection.y, viewDirection.z)
    near.d = -near.a*(eye.x+viewDirection.x*nearDistance)
         -near.b*(eye.y+viewDirection.y*nearDistance)
         -near.c*(eye.z+viewDirection.z*nearDistance)

    (far.a, far.b, far.c) =
      (-viewDirection.x,-viewDirection.y, -viewDirection.z)
    far.d = -far.a*(eye.x+viewDirection.x*farDistance)
         -far.b*(eye.y+viewDirection.y*farDistance)
         -far.c*(eye.z+viewDirection.z*farDistance)

    tangent = tan(opennessAngle/2.);
    farY = tangent * farDistance;
    farZ = tangent * farDistance;

    left = createPlaneFromPoints(0, 0, 0, farDistance, farY, -farZ,
        farDistance, farY, farZ)
    right = createPlaneFromPoints(0, 0, 0, farDistance, -farY, farZ,
        farDistance, -farY, -farZ)
    top = createPlaneFromPoints(0, 0, 0, farDistance, farY, farZ,
        farDistance, -farY, farZ);
    bottom = createPlaneFromPoints(0, 0, 0, farDistance, -farY, -farZ,
        farDistance, farY, -farZ)
  end
end
```

B.2.2.1/ ALGORITHME DE CALCUL DE L'ENSEMBLE DES OBJETS PERÇUS

```
class FrustumCuller implements Iterator
begin
  // Holds the frustum used for the culling
  frustum : Frustum

  // tree on which the iteration must be applied.
  tree : Tree

  // STACK of nodes that are encountered during the iteration but not
  // yet treated
  availableNodes : Stack of TreeNode

  // Initialise the iterator
  function initialise()
  begin
    if frustum.intersect(tree.root.box) availableNodes += tree.root;
  end

  // Replies if a element is available in the iterator
  function hasNext()
  begin
    return ! availableNodes.isEmpty()
  end

  // Replies the next available element from this iterator
  function next()
  begin
    // Test if a node is available
    if (availableNodes.isEmpty()) return None;

    // Extract the next available node
    current = availableNodes.top();

    // Add the child nodes as available nodes
    foreach child from current.getChildrenNodes()
      if frustum.intersect(child.box) availableNodes += child;
    end for

    return current;
  end
end
```

C

FICHIER DE CONFIGURATION JASIM

Ce chapitre est extrait des documents de spécification de JASIM, écrites initialement en anglais.

The main goal of JANUS platform is to facilitate multi-agent technology transfers and the use of an organisational and holonic approach in industrial projects. Holonic multi-agent systems introduce selfsimilar and recursive entities. To be implemented holonic models require a platform able to manage the concept of hierarchy of composition between entities : JANUS. The heart of the platform provides an implementation of the CRIO organisational metamodel [Rodríguez et al., 2007] and the concept of holon. Platform core is based on a micro-kernel architecture which allows distribution on a network. JANUS platform is generic and extensible. It is entirely developed in Java and uses JXTA technologies for a part of network support.

Simulation of autonomous entities in an complex urban system requires dedicated software models. JASIM platform integrates components which are required to simulation complex environments in 3D (in particular used for simulation in virtual reality). This platform integrates several models to reproduce human visual perception in a virtual environment and endogenous behavior of this environment. Thus simulated entities can used the JASIM platform to perceive and act in a situated system. JASIM platform also provides networking tools which are allowing a visualization software to have suitable informations on simulated environment's state. JASIM platform is entirely written in Java.

This chapter is dedicated to the specification of simulation configuration file which may be used to describe simulation scenarios.

C.1/ DESCRIPTION

JASIM simulation may run specific scenarios. To avoid hard-coded scenario, JASIM is able to read simulation configuration files (aka. SFG file). SFG file is an XML file which is compliant with one of the Document Type Definitions (DTD) described in the rest of this chapter.

The purpose of SFG files is to provide to JASIM informations to load a simulated world, spawn entities, create probes, etc.

Because a $3D$ simulation has not the same configuration information as for a $1.5D$ simulation, a DTD is provided for each of simulation type.

C.2/ Type Definition

C.2.1/ String

A string is a sequence of characters. If characters quote (") or backslash (\) may be inside string, they must be preceeded by a backslash (\).

C.2.2/ UUID

A Universally Unique Identifier (UUID) is an identifier standard used in software construction, standardized by the Open Software Foundation (OSF) as part of the Distributed Computing Environment (DCE). The intent of UUIDs is to enable distributed systems to uniquely identify information without significant central coordination. Thus, anyone can create a UUID and use it to identify something with reasonable confidence that the identifier will never be unintentionally used by anyone for anything else. Information labeled with UUIDs can therefore be later combined into a single database without needing to resolve name conflicts.

UUIDs are documented as part of ISO/IEC 11578:1996 “Information technology — Open Systems Interconnection — Remote Procedure Call (RPC)” and more recently in ITU-T Rec. X.667 | ISO/IEC 9834-8 :2005. The IETF has published Proposed Standard RFC 4122 that is technically equivalent with ITU-T Rec. X.667 | ISO/IEC 9834-8.

A UUID is a 16-byte (128-bit) number. The number of theoretically possible UUIDs is therefore about 3×1038. In its canonical form, a UUID consists of 32 hexadecimal digits, displayed in 5 groups separated by hyphens, in the form $8-4-4-4-12$ for a total of 36 characters (32 digits and 4 hyphens). For example : `550e8400-e29b-41d4-a716-446655440000`.

C.2.3/ Date

Date are specific-format strings which corresponds to a day in the gregorian calendar. Supported formats are :
– Every format supported by Java `DateFormat` class ;
– format `yyyy-MM-dd` ;
– format `yyyy/MM/dd` ;
– format `dd/MM/yyy`.

C.2.4/ Integer

An integer is a string which is representing a integer number. Leading and trailing whitespace characters in string are ignored. The sequence of characters following an (optional) negative sign and/or radix specifier (`"0x"`, `"0X"`, `"#"`, or leading zero) is parsed as with one of the radix (10, 16, or 8). This sequence of characters must represent a positive

value. The result is negated if first character of the specified string is the minus sign. No whitespace characters are permitted in the string.

Accepts decimal, hexadecimal, and octal numbers numbers given by the following grammar :

Integer :	Sign DecimalNumeral
	Sign `0x` HexDigits
	Sign `0X` HexDigits
	Sign `#` HexDigits
	Sign `0` OctalDigits
Sign :	`[-+]?`
DecimalNumeral :	`[0-9]+`
HexDigits :	`[0-9a-eA-E]+`
OctalDigits :	`[0-7]+`

C.2.5/ FLOAT

A float is a string which is representing a double precision floating point number. Leading and trailing whitespace characters in string are ignored. The rest of string should constitute a float value as described by the lexical rule :

Float :	Sign `NaN`
	Sign `Infinity`
	Sign ScientificLiteral
	Sign FloatingPointLiteral
Sign :	`[-+]?`
ScientificLiteral :	FloatNumeral `e` Sign DecimalNumeral
FloatingPointLiteral :	FloatingNumeral SinglePrecisionType
DecimalNumeral :	`[0-9]+`
FloatNumeral :	`[0-9]+.[0-9]+`
	`[0-9]+.`
	`.[0-9]+`
SinglePrecision :	`[fd]?`

C.2.6/ BOOLEAN

A boolean is a string which is representing a boolean value. True value is representing by string `"true"` (case insensitive), and false value is representing by string `"false"` (also case insensitive).

C.2.7/ URL

URL represents a Uniform Resource Locator, a pointer to a “resource” on the World Wide Web. A resource can be something as simple as a file or a directory, or it can be a reference to a more complicated object, such as a query to a database or to a search engine. More information on the types of URLs and their formats can be found at : `ftp://ftp.ncsa.uiuc.edu/Mosaic/Papers/url-primer.ps.Z`.

In general, a URL can be broken into several parts. The previous example of a URL indicates that the protocol to use is `http` (HyperText Transfer Protocol) and that the information resides on a host machine named `www.ncsa.uiuc.edu`. The information on that host machine is named `/SDG/Software/Mosaic/Demo/url-primer.html`. The exact meaning of this name on the host machine is both protocol dependent and host dependent. The information normally resides in a file, but it could be generated on the fly. This component of the URL is called the path component.

Major supported protocols are :
– `http`, supported by default by Java virtual machine ;
– `ftp`, supported by default by Java virtual machine ;
– `file`, supported by default by Java virtual machine ;
– `resource`, not supported by Java virtual machine. It represents an absolute path to a Java resource (see `Class.getResource()` manual page for more details on resources) ;
– “no protocol”, it is a special case where the given string represents the path part of an URL. It will be used to build an absolute path by the readers.

C.2.8/ CLASSNAME

This type corresponds to the name of a Java class, according to the Java specifications.

C.2.9/ TIME

Type Time is a specific type of Float. It corresponds to a simulation time. It is an positive floating-point number.

C.2.10/ VARIANT

Type Variant is a string which is interpreted depending on its content. Any reader supporting this type may try to create objects corresponding to the string. Classic Java implementation should try the following matchings :

1. if the given string could be parsed as a date-time pair, a `java.util.Date` instance is used ;
2. if the given string could be parsed as a boolean, a `java.lang.Boolean` instance is used ;
3. if the given string could be parsed as a floating-point number, a `java.lang.Double` instance is used ;
4. if the given string could be parsed as an integer number, a `java.lang.Long` instance is used ;
5. othewise the string value is used.

C.3/ TAG DESCRIPTION

This section describes all the tags available for SFG file. The root tag is <simulation/> .

C.3.1/ <SIMULATION/>

Tag <simulation/> is describing a simulation scenario. Tag <environment/> is mandatory, and <time/> and <spawners/> tags are not. Supported attributes are listed in table C.1.

Name	*Type*	*Read-only*	*Mandatory*	*Default value*
dtdversion	String	yes	yes	7.0
id	UUID	no	yes	
name	String	no	no	
date	Date	no	no	today
authors	String	no	no	
version	String	no	no	
description	String	no	no	

TABLE C.1 – <simulation/> Attributes

Attribute dtdversion must contains the same version number as the supported DTD. It is not required to put this attribute in the XML files because it is automatically deduced from the DTD itself. The simulation scenario has an unique id, a name, a last-change date, authors and a version number. All these attributes are optional. Finally a simulation may be described with a short text in the description attribute.

C.3.2/ <TIME/>

Tag <time/> may appear in tag <simulation/> . It describes how JASIM is supposed to manage time during the simulation. Tag <time/> contains only attributes, described in table C.2. Default values are underlyed.

Name	*Type*	*Read-only*	*Mandatory*	*Values*
type	String	no	yes	step, realtime
unit	String	no	yes	millisecond, second, minute, hour
timeStep	Float	no	yes	$]0; +\infty[$, 1

TABLE C.2 – <time/> Attributes

Time manager is invoked by JASIM simulator at each step of the simulation loop. When step type, time evolves by a constant amount given by timeStep. When realtime type, time evolves according to the operating system clock. In both cases unit is mandatory to determine which is the time unit in the simulation.

C.3.3/ <ENVIRONMENT/>

Environment is a major part of JASIM simulation. It is the entity in which agents are lying and living. It provides perceptions to agents and accepts actions from them. In JASIM simulation, environment is composed of places. Place is a part of the environment which could be run independently from other places. Simulated entities (aka. agents) are able to move from place to place when they traverse portals. Tag <environment/> is composed

of tag <places/> and optionally tag <portals/> . <environment/> tag contains only one read-only attribute : dimension. It indicates the supported dimension by the environment model : $3d$ or $1.5d$.

C.3.4/ <PLACES/> AND <PLACE/>

Place is a small part of environment which could be run indenpently from other places. Tag <places/> does not contain attribute. It contains a tag <place/> for each place in the environment. Tag <place/> is composed of four subtags : <groundEnvironment/> , <staticEnvironment/> , <dynamicEnvironment/> , and <environmentProbes/> . Theses tags are describing resctively ground, immobile entities, mobile entities in an environment, and environment probes to load. Table C.3 describes the attributes of tag <place/> .

Name	*Type*	*Read-only*	*Mandatory*	*Values*
activate	Boolean	no	yes	true, false
id	UUID	no	yes	
name	String	no	no	
perceptionAlgo	String	no	no	sequential, parallel
perceptionTraversal	String	no	no	topdown, bottomup

TABLE C.3 – <place/> Attributes

Attribute activate indicates if JASIM is able to load the place for simulation. Attribute id is important because it may be referenced by many other tags of SFG file.

The tag <place/> also enable to specify the type of algorithm to run for the computation of the agent perceptions. Attribute perceptionAlgo describes the run type of the algorithm : sequential execution forces the algorithm to run the perception computation for each agent in turn ; and parallel execution allows the algorithm to run the perception computation for each agent on different threads. Additionally the attribute perceptionTraversal enables to force how the perception algorithm will traverse the data structures : the top-down approach means that the data-structure is traversed from the world to the entities ; and the bottom-up approach means that the data-structure is traversed from the entity to the world.

C.3.5/ <GROUNDENVIRONMENT/>

Tag <groundEnvironment/> permits to describes properties for the place's ground. Ground is a key component of a place. It permits to compute heights of entities in $3D$ environments or to describe road networks in $1.5D$ environments. Table C.4 lists the attributes of tag <groundEnvironment/> .

When $3D$, type of ground defines how JASIM will react on ground queries :
- if type is alignedIndoor or orientedIndoor, the ground is a plane (as many indoor grounds). In this case tag <groundEnvironment/> must contain a tag <indoorGround/> ;
- if type is heightmap or repulsionHeightmap, the ground is not a plane and each height is defined in a heightmap. In this case tag <groundEnvironment/> must contain a tag <heightmap/> ;

Dim.	*Name*	*Type*	*Read-only*	*Mandatory*	*Values*
all	id	UUID	no	yes	
$3D$	type	String	no	yes	alignedIndoor, orientedIndoor, heightmap, repulsionHeightmap, any
$3D$	prebuildResource	URL	no	no	
$1.5D$	shapeFile	URL	no	yes	
$1.5D$	dbaseFile	URL	no	no	

TABLE C.4 – <groundEnvironment/> Attributes

– if type is any, the ground type depends on the child tag. If <indoorGround/> is a child of <groundEnvironment/> then ground is assumed to be of type alignedIndoor or orientedIndoor according to <indoorGround/> content. If <heightmap/> is a child of <groundEnvironment/> then ground is assumed to be of type heightmap.

Types heightmap and repulsionHeightmap are both based on a heightmap. A heightmap is a bitmap in which the colors define the height of each point on the ground. Both types use red-components to retreive the heights. Repulsion ground uses the green- and blue-components as the two components of a repulsion vector. A repulsion vector is a vector which indicates a repulsive force from an obstacle or a not traversable area.

Attribute prebuildResource indicates the URL of a serialized instance of the Java Ground interface. If provided, the given resource will be deserialized and directly pass to JASIM as Ground object. If not provided, the Ground object will be created according to the <groundEnvironment/> declaration.

When $1.5D$ environment, attribute shapeFile is the path to geometries of a road network. The referenced file must be an ESRI Shape file. Attribute dbaseFile is the path to attributes associated to geometries. The referenced file must be an dBase file. When $1.5D$ environment, tag <groundEnvironment/> has no child.

C.3.6/ <STATICENVIRONMENT/>

Tag <staticEnvironment/> describes the static/immobile objects which are in the simulated world. This tag does not take any child but has only attributes described in table C.5.

Name	*Type*	*Read-only*	*Mandatory*	*Values*
class	Classname	no	yes	
prebuildResource	URL	no	no	

TABLE C.5 – <staticEnvironment/> Attributes

The given classname permits to create an empty data structure. For $3D$ environment attribute prebuildResource is the URL of a serialized instance of a static perception tree which has the given classname as type.

For $1.5D$ environment tag <staticEnvironment/> is not used because it is included in ground model.

C.3.7/ <DYNAMICENVIRONMENT/>

Tag <dynamicEnvironment/> describes the mobile objects which are in the simulated world at startup. This tag takes a set of <agent/> tags as children and attributes described in table C.6.

Dim.	*Name*	*Type*	*Read-only*	*Mandatory*	*Values*
all	class	Classname	no	yes	
all	prebuildResource	URL	no	no	
$3D$	agentType	Classname	no	no	

TABLE C.6 – <dynamicEnvironment/> Attributes

The given classname permits to create an empty data structure. For $3D$ environment attribute prebuildResource is the URL of a serialized instance of a static perception tree which has the given classname as type. If given, attribute agentType specifies the default Java class which may be used to instance agents at startup. This attribute may be overridden by <agent/> child tags.

For $1.5D$ environment, attributes class and prebuildResource are ignored. But subtags <agent/> are parsed.

C.3.8/ <AGENT/>

Tag <agent/> describes how to create an instance of agent for a simulation scenario. Each instance of <agent/> indicates a mapping between a agent's java class and a body type. Attributes of <agent/> are listed in table C.7. Additional semantics may be attached to agents for all the <agentSemantic/> children in <agent/> .

Name	*Type*	*Read-only*	*Mandatory*	*Values*
activate	Boolean	no	yes	true, false
class	Classname	no	yes	
type	Classname	no	yes	

TABLE C.7 – <agent/> Attributes

Attribute activate indicates if the mapping described by the <agent/> tag is enabled or not. Attribute class must contain a Java classname of an agent. This type is used to create agent's instance. Attribute type must contain a Java classname of body type. It is used to create a body inside the environment for the agent's instance. The classname specified in attribute type must be a sunclass of `fr.utbm.set.jasim.environment.semantics.BodyType`.

C.3.9/ <AGENTSEMANTICS/> AND <AGENTSEMANTIC/>

Tag <agentSemantics/> contains a list of <agentSemantic/> . Tag <agentSemantics/> constains as raw text a java classname which is describing a semantic (subclass of `fr.utbm.set.jasim.environment.semantics.Semantic`.

C.3.10/ <environmentProbes/> and <environmentProbe/>

Tag <environmentProbes/> contains a list of <environmentProbe/> . Only one attribute is available in <environmentProbes/> (see table C.8). This attribute activate indicates if the set of probes is enabled or not.

Name	*Type*	*Read-only*	*Mandatory*	*Values*
activate	Boolean	no	yes	true, false

TABLE C.8 – <environmentProbes/> Attributes

Tag <environmentProbe/> describes a probe in the environment. Environment probes permit to retreive information and data from places without any influence on the place's behaviours. Table C.9 describes the <environmentProbe/> attributes.

Dim.	*Name*	*Type*	*Read-only*	*Mandatory*	*Values*
all	activate	Boolean	no	yes	true, false
all	name	Classname	no	yes	
all	x	Float	no	no	$]-\infty;+\infty[$, 0
all	y	Float	no	no	$]-\infty;+\infty[$, 0
$3D$	z	Float	no	no	$]-\infty;+\infty[$, 0

TABLE C.9 – <environmentProbe/> Attributes

Probe may be activated or not. It may have a position according to attributes x, y, and optionally z. And finally a probe must have a name which is corresponding to the Java classname of the probe implementation.

C.3.11/ <portals/> and <portal/>

Tag <portals/> contains a list of <portal/> . Only one attribute is available in <portals/> (see table C.10). This attribute activate indicates if the set of portals is enabled or not.

Name	*Type*	*Read-only*	*Mandatory*	*Values*
activate	Boolean	no	yes	true, false

TABLE C.10 – <portals/> and <portal/> Attributes

Tag <portal/> describes a mono-directional portal from one place to another. A portal has one attribute described in table C.10. Tag <portal/> has two mandatory children : <portalSource/> and <portalTarget/> . They describe where the portal starts and terminates respectively.

C.3.12/ <portalSource/>

Tag <portalSource/> describes where a portal is starting from a place. Table C.11 lists the available attributes.

Dim.	*Name*	*Type*	*Read-only*	*Mandatory*	*Values*
all	place	UUID	no	yes	
all	y1	Float	no	no	$]-\infty;+\infty[$, $\underline{0}$
all	y2	Float	no	no	$]-\infty;+\infty[$, $\underline{0}$
$3D$	x1	Float	no	no	$]-\infty;+\infty[$, $\underline{0}$
$3D$ x2	Float	no	no	$]-\infty;+\infty[$, $\underline{0}$	
$3D$	onleft	Boolean	no	no	true, false
$1.5D$	road	UUID	no	yes	
$1.5D$	x	Float	no	no	$]-\infty;+\infty[$, $\underline{0}$
$1.5D$	direction	String	no	no	segment, reverse, both

TABLE C.11 – <portalSource/>

A portal source is on the place identified by attribute place.

Location of the portal source is represented by a segment on the place. For $3D$ environment attributes x1, y1, x2, y2 are the coordinates of the segment points. For $1.5D$ environment attribute road is the identifier of the road segment where the portal is located ; attribute x is the curviline coordinate of the portal on the road ; and attributes y1, y2 are the minimal and maximal shifting coordinates of the portals.

The portal is traversable only on one side of the segment. For $3D$ environment attribute onleft indicates the side which is traversable. If attribute onleft is `true` (resp. `false`), the portal is traversable only for entities whic are crossing the portal segment from left to right (resp. right to left). The left/right sides are determined by the vector $(x2;y2)-(x1;y1)$.

For $1.5D$ environment attribute direction indicates on which direction the portal is traversable : on the segment direction, on the reverse direction, or on both directions.

C.3.13/ <PORTALTARGET/>

Tag <portalTarget/> describes where an entity traversing the portal lyies in the target place. Table C.12 lists the available attributes.

Dim.	*Name*	*Type*	*Read-only*	*Mandatory*	*Values*
all	place	UUID	no	yes	
all	x	Float	no	no	$]-\infty;+\infty[$, $\underline{0}$
all	y	Float	no	no	$]-\infty;+\infty[$, $\underline{0}$
$3D$	dx	Float	no	no	$]-\infty;+\infty[$, $\underline{0}$
$3D$	dy	Float	no	no	$]-\infty;+\infty[$, $\underline{0}$
$1.5D$	direction	String	no	no	segment, reverse

TABLE C.12 – <portalTarget/> Attributes

A portal target is a location in the place identified by attribute place. Portal target is always a point located at (x,y) in a $3D$ environment, and at $(road,x,y)$ in a $1.5D$ environment. Every entity traversing the portal is moved on the given point and oriented according to the given direction : (dx,dy) for $3D$ and direction for $1.5D$. Direction (dx,dy) will not becomes the orientation of the entities. It represents the output direction of the portal. If an entity

is traversing a portal, its orientation may be preserved. In other words, the difference between the entity orientation before traversing and the portal source's segment orientation may be the same as the difference between the entity orientation after traversing and (dx, dy).

C.3.14/ <HEIGHTMAP/>

Tag <heightmap/> describes heights of a ground inside a bitmap. Because bitmap is in pixel space, <heightmap/> permits to put and scale the bitmap in the simulation world. Several parts of a ground are not traversable : hole, sea, obstacles... The lowest traversable height is named "ground zero". It is represented by a color component in one hand, and a height in world coordinate system in other hand. Table C.13 describes attributes of <heightmap/> .

Name	*Type*	*Read-only*	*Mandatory*	*Values*
url	URL	no	yes	
minx	Float	no	yes	$]-\infty;+\infty[/minx \leq maxx$
miny	Float	no	yes	$]-\infty;+\infty[/miny \leq maxy$
minz	Float	no	yes	$]-\infty;+\infty[/minz \leq maxz$
maxx	Float	no	yes	$]-\infty;+\infty[/minx \leq maxx$
maxy	Float	no	yes	$]-\infty;+\infty[/miny \leq maxy$
maxz	Float	no	yes	$]-\infty;+\infty[/minz \leq maxz$
rawGroundZero	Integer	no	no	$]-128;127[, \underline{-127}$
groundZero	Float	no	no	$]minz;maxz[$
semantic	Classname	no	no	

TABLE C.13 – <heightmap/> Attributes

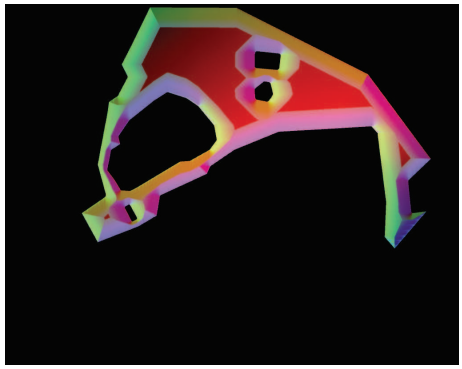
Attribute url is the location of the heightmap file. Figure C.1(a) illustrates a bitmap in gray scale. Figure C.1(a) illustrates the same heightmap with repulsive vectors encoded in green- and blue-components. Figure C.1(c) illustrate a view of the repulsive vectors. Attributes minx and miny are the world coordinates of the lowest point covered by the ground. Attributes maxx and maxy are the world coordinates of the uppest point covered by the ground. Attributes minz and maxz correspond to the minimal and maximal heights in world coordinate system. Attributes rawGroundZero and groundZero are representing the height under which the ground is not traversable. Attribute rawGroundZero is expressed in pixel color and groundZero in height in world model. Finally it is possible to associate a semantic to the ground. This semantic may be a subclass of `fr.utbm.set.jasim.environment.semantics.GroundSemantic`.

C.3.15/ <INDOORGROUND/>

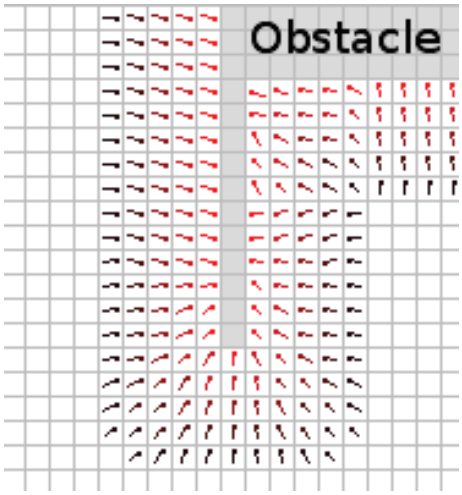
Tag <indoorGround/> describes a ground which is planar. This tag may contain a <alignedArea/> tag or <orientedArea/> . Both child tags describe the area covered by the indoor ground. Common attributes are listed in table C.14.

Attribute z corresponds to the height of the ground in world coordinate system. Finally it is possible to associate a semantic to the ground. This semantic may be a subclass of

(a) Heightmap with Heights Only



(b) Heightmap with Heights and Repulsive Vectors



(c) View of Repulsive Vectors

FIGURE C.1 – Heighmap Examples

`fr.utbm.set.jasim.environment.semantics.GroundSemantic`.

C.3.16/ <ALIGNEDAREA/> AND <ORIENTEDAREA/>

Tag <alignedArea/> describes a planar area which has its borders aligned on the world axis. Attributes are listed in table C.15.

Attributes minx and miny are the world coordinates of the lowest point covered by the area. Attributes maxx and maxy are the world coordinates of the uppest point covered by the area.

Tag <orientedArea/> describes a planar area which has its borders not-always aligned on the world axis. Attributes are listed in table C.16.

The oriented area is described by an oriented bounding rectangle of center point located at the position given by attributes centerx and centey. The two directions of the rectangle (R and S vectors) are defined by two vectors for which the numerical components are give nby attributes Rx, Ry, Sx, and Sy.

Name	Type	Read-only	Mandatory	Values
z	Float	no	yes	$]-\infty;+\infty[/minz \leq maxz$
semantic	Classname	no	no	

TABLE C.14 – <indoorGround/> Attributes

Name	Type	Read-only	Mandatory	Values
minx	Float	no	yes	$]-\infty;+\infty[/minx \leq maxx$
miny	Float	no	yes	$]-\infty;+\infty[/miny \leq maxy$
maxx	Float	no	yes	$]-\infty;+\infty[/minx \leq maxx$
maxy	Float	no	yes	$]-\infty;+\infty[/miny \leq maxy$

TABLE C.15 – <alignedArea/> Attributes

C.3.17/ <SPAWNERS/> AND <SPAWNER/>

Tag <spawners/> contains a list of <spawner/> . Only one attribute is available in <spawners/> (see table C.17). This attribute activate indicates if the set of portals is enabled or not.

Tag <spawner/> describes an area where agents may be spawned. Table C.18 describes the attributes of <spawner/> . This tag may contains several <entity/> tags to describe how to spawn entities.

A spawner may be activated or not. If it was activated, spawning occurs when startDate is reached and not endDate (if given). Attributes id and names are not yet used to reference spawners in JASIM. Attribute place may contains the identifier of the place where the spawner is located, additionally for $1.5D$ environment attribute road indicates the road segment where the spawner is located.

A spawner must be of two types : point or area. When spawner is a point only attributes x, y (and z for $3D$) are used ; not attributes width (and height for $3D$). When spawner is an area attributes x, y (and z for $3D$) are coordinates of the lowest point of the spawning area and attributes width (and height for $3D$) are the size(s) of the spawning area.

C.3.18/ <ENTITY/>

Tag <entity/> describes how to spawn an entity in the simulated world. Table C.19 describes <entity/> attributes. Tags <frustums/> (mandatory), <body/> (optional), <agentSemantics/> (optional), <waypoints/> (optional), <goals/> (optional), <generationLaw/> (mandatory), <attributes/> (optional), and <probes/> (optional) may children of <entity/> .

Entity spawning could be activated or not with attribute activate. Attribute agentType is the Java classname which may be created to spawn an agent (see section C.3.18.1 for details). If given, attribute budget indicates how many agents will be spawned.

Name	*Type*	*Read-only*	*Mandatory*	*Values*
centerx	Float	no	yes	$]-\infty;+\infty[$
centery	Float	no	yes	$]-\infty;+\infty[$
Rx	Float	no	yes	$]-\infty;+\infty[$
Ry	Float	no	yes	$]-\infty;+\infty[$
Sx	Float	no	yes	$]-\infty;+\infty[$
Sy	Float	no	yes	$]-\infty;+\infty[$

TABLE C.16 – <orientedArea/> Attributes

Name	*Type*	*Read-only*	*Mandatory*	*Values*
activate	Boolean	no	yes	true, false

TABLE C.17 – <spawners/> Attributes

C.3.18.1/ AGENT SPAWNING CONSTRAINTS

Agent are spawned by creating an instance of the given Java class (attribute agentType of <entity/>). To be usable the given class may implement one or more of the following constructors. The constructors are searched by the spawner in the given order :

1. `Classname(List<? extends WayPoint>, List<? extends GoalPoint>, Frustum<?,?>, Map<String,Object>)`
2. `Classname(List<? extends WayPoint>, List<? extends GoalPoint>, Iterable<Frustum<?,?>>, Map<String,Object>)`
3. `Classname(List<? extends WayPoint>, List<? extends GoalPoint>, Map<String,Object>)`
4. `Classname(List<? extends WayPoint>, List<? extends GoalPoint>, Frustum<?,?>)`
5. `Classname(List<? extends WayPoint>, List<? extends GoalPoint>, Iterable<Frustum<?,?>>)`
6. `Classname(List<? extends WayPoint>, List<? extends GoalPoint>)`
7. `Classname(List<? extends GoalPoint>, Map<String,Object>)`
8. `Classname(List<? extends GoalPoint>, Frustum<?,?>)`
9. `Classname(List<? extends GoalPoint>, Iterable<Frustum<?,?>>)`
10. `Classname(Map<String,Object>)`
11. `Classname(List<? extends GoalPoint>)`
12. `Classname(Frustum<?,?>)`
13. `Classname(Iterable<Frustum<?,?>>)`
14. `Classname()`

Parameters of these contructors will take there values according to :
– `List<? extends WayPoint>` is the list of waypoints given by tag <waypoints/> ;
– `List<? extends GoalPoint>` is the list of goals given by tag <goals/> ;
– `Frustum<?,?>` is an instance of frustum given by the first occurence of <frustum/> in <frustums/> tag ;

Dim.	*Name*	*Type*	*Read-only*	*Mandatory*	*Values*
all	activate	Boolean	no	yes	true, false
all	id	UUID	no	yes	
all	name	String	no	no	
all	place	UUID	no	yes	
all	type	String	no	yes	point, area
all	x	Float	no	no	$]-\infty;+\infty[$, 0
all	y	Float	no	no	$]-\infty;+\infty[$, 0
all	width	Float	no	no	$[0;+\infty[$, 0
all	startDate	Time	no	no	$[0;+\infty[$, 0
all	endDate	Time	no	no	$]startDate;+\infty[$
$3D$	z	Float	no	no	$]-\infty;+\infty[$, 0
$3D$	height	Float	no	no	$[0;+\infty[$, 0
$1.5D$	road	UUID	no	yes	
$1.5D$	direction	String	no	no	segment, reverse, both

TABLE C.18 – <spawner/> Attributes

Name	*Type*	*Read-only*	*Mandatory*	*Values*
activate	Boolean	no	yes	true, false
agentType	Classname	no	yes	
budget	Integer	no	no	$[0;+\infty[$, $+\infty$

TABLE C.19 – <entity/> Attributes

– `Iterable<Frustum<?,?>>` is the list of frustums given by tag <frustums/> ;
– `Map<String,Object>` is the collection of ⟨identifier, value⟩ pairs given by tag <attributes/> .

C.3.19/ <FRUSTUMS/> AND <FRUSTUM/>

Tag <frustums/> contains a list of <frustum/> . No attribute is available in <frustums/> . Tag <frustum/> permits to describe perception frustum for spawned entities. Table C.20 contains the XML attributes of this tag.

Attribute eyePosition is the vertical distance between the position of the entity and eye position. By default, the position of the entity is located on the ground and eyes are at 1.8 meters high.

Frustum has different types (illustrated by figure C.2) :
– “sphere” : frustum is a sphere centered on the eye position with a radius given by farDistance ;
– “pyramidal” : frustum is a truncated pyramidal composed of four planes. Far and near planes are perpendicular to view direction and located at farDistance and nearDistance meters respectively. Left, right, top and bottom planes are intersecting at eye position. Left and right (resp. top and bottom) planes have an constant angle given by attribute fieldOfViewHAngle (resp. fieldOfViewVAngle). These four planes are located around view direction for form a pyramid with far plane.

Name	*Dim.*	*Type*	*Read-only*	*Mandatory*	*Values*
$3D$	type	String	no	yes	rectangle, sphere, pyramid, pedestrian
$3D$	eyePosition	Float	no	no	1.8
$3D$	farDistance	Float	no	yes	$[0; +\infty[$
$3D$	sideSize	Float	no	no	$[0; +\infty[$, $\underline{0}$
$3D$	verticalSize	Float	no	no	$[0; +\infty[$, $\underline{0}$
$3D$	nearDistance	Float	no	no	$[0; farDistance]$, $\underline{0}$
$3D$	fieldOfViewHAngle	Float	no	no	$[-\infty; +\infty[$, $\underline{\pi/4}$
$3D$	fieldOfViewVAngle	Float	no	no	$[-\infty; +\infty[$, $\underline{\pi/4}$
$1.5D$	type	String	no	yes	rectangle
$1.5D$	farDistance	Float	no	yes	$[0; +\infty[$
$1.5D$	sideSize	Float	no	yes	$[0; +\infty[$

TABLE C.20 – <frustum/> Attributes

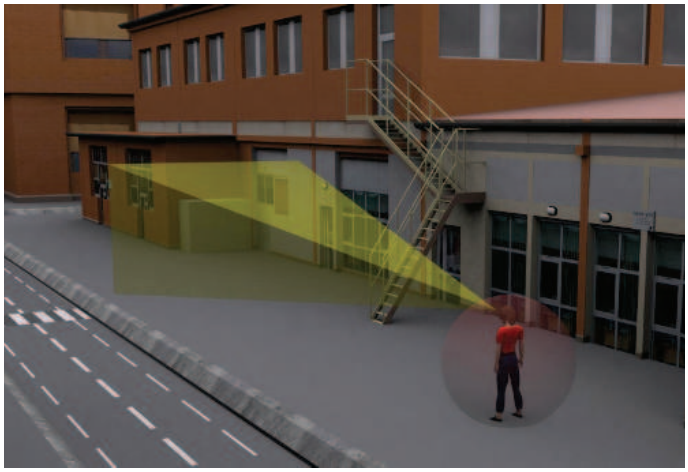


FIGURE C.2 – Example of Pedestrian Frustum, composed of a Spherical Frustum and a Pyramidal Frustum

– “pedestrian” : frustum is the composition of a spherical frustum and a pyramidal frustum. Psherical frustum is centered on the eye position with radius given by attribute nearDistance. Pyramidal frustum uses same attributes as previously.
– “rectangle” : frustum is parallelepiped box where eye is located at the center of the base side of the parallelepiped. Size from left to right planes is given by attribute sideSize. Size from eye position to far place is given by attribute farDistance. Size from top to bottom planes is goven by attribute verticalSize.

C.3.20/ <BODY/>

Tag <body/> describes the geometrical property of agent’s bodies. Table C.21 describes attributes.

Type of <body/> is a subclass of `fr.utbm.set.jasim.environment.semantics.BodyType`. All bodies are assumed to be cylinders in JASIM $3D$ environment and rectangular in $1.5D$

Dim.	*Name*	*Type*	*Read-only*	*Mandatory*	*Values*
all	type	Classname	no	no	
$3D$	height	Float	no	no	
$3D$	radius	Float	no	no	
$1.5D$	curvilineSize	Float	no	no	
$1.5D$	lateralSize	Float	no	no	

TABLE C.21 – <body/> Attributes

environment. Attributes height and radius describes the size of the cylinder. Attribute curvilineSize and lateralSize describes the size of the rectangle.

C.3.21/ <WAYPOINTS/> AND <WAYPOINT/>

Tag <waypoints/> contains a list of <waypoint/> . Only one attribute is available in <waypoints/> (see table C.22). This attribute activate indicates if the set of waypoints is enabled or not.

Name	*Type*	*Read-only*	*Mandatory*	*Values*
activate	Boolean	no	yes	true, false

TABLE C.22 – <waypoints/> Attributes

Tag <waypoint/> describes a point in world on which agents may pass. Table C.23 describes the attributes of <waypoint/> .

Dim.	*Name*	*Type*	*Read-only*	*Mandatory*	*Values*
all	activate	Boolean	no	yes	true, false
all	name	String	no	no	
all	place	UUID	no	yes	
all	x	Float	no	no	0
all	y	Float	no	no	0
all	velocity	Time	no	no	
all	time	Time	no	no	
$3D$	z	Float	no	no	0
$3D$	tangentX	Float	no	no	
$3D$	tangentY	Float	no	no	
$3D$	tangentZ	Float	no	no	
$1.5D$	road	UUID	no	yes	
$1.5D$	direction	String	no	no	segment, reverse, both

TABLE C.23 – <waypoint/> Attributes

<waypoint/> may be activated to be passed to JASIM.

For $3D$ environment attributes x, y, and z are the location of the waypoint. Attributes tangentX, tangentY, tangentZ are the components of desired view vector for agents when they reach the waypoint.

For $1.5D$ environment attributes road, x, y are respectively the identifier of the road where is located the waypoint, the curviline position, and the lateral position of the waypoint on the road.

Attribute velocity is the desired speed of agents when they reach the waypoint. Attribute time is the desired time at which the waypoint may be reached by agents.

C.3.22/ <GOALS/> AND <GOAL/>

Tag <goals/> contains a list of <goal/> . Only one attribute is available in <goals/> (see table C.24). This attribute activate indicates if the set of goals is enabled or not.

Name	*Type*	*Read-only*	*Mandatory*	*Values*
activate	Boolean	no	yes	true, false

TABLE C.24 – <goals/> Attributes

Tag <goal/> describes a point in world which agents may reach. Table C.25 describes the attributes of <goal/> .

Dim.	*Name*	*Type*	*Read-only*	*Mandatory*	*Values*
all	activate	Boolean	no	yes	true, false
all	name	String	no	no	
all	place	UUID	no	yes	
all	x	Float	no	no	0
all	y	Float	no	no	0
all	time	Time	no	no	
$3D$	z	Float	no	no	0
$3D$	tangentX	Float	no	no	
$3D$	tangentY	Float	no	no	
$3D$	tangentZ	Float	no	no	
$1.5D$	road	UUID	no	yes	
$1.5D$	direction	String	no	no	segment, reverse, both

TABLE C.25 – <goal/> Attributes

<goal/> may be activated to be passed to JASIM. For $3D$ environment attributes x, y, and z are the location of the goal. Attributes tangentX, tangentY, tangentZ are the components of desired view vector for agents when they reach the goal. For $1.5D$ environment attributes road, x, y are respectively the identifier of the road where is located the goal, the curviline position, and the lateral position of the goal on the road.

Attribute time is the desired time at which the goal may be reached by agents.

C.3.23/ <GENERATIONLAW/> AND <LAWPARAM/>

Tag <generationLaw/> describes stochastic law to generate agents. Table C.26 describes the attributes of <generationLaw/> .

Name	Type	Read-only	Mandatory	Values
class	Classname	no	yes	

TABLE C.26 – <generationLaw/> Attributes

Attribute class is the name of the Java class which is implementing the stochastic law. The known stochastic laws and associated parameters are described into the full specification document. These parameters may be given by <lawParam/> subtags. Table C.27 describes the attributes of <lawParam/> . Attribute name is the name of the law's parameter, and attribute value is the floating-point number value to pass to the generation law.

Name	Type	Read-only	Mandatory	Values
name	String	no	yes	
value	Float	no	yes	

TABLE C.27 – <lawParam/> Attributes

C.3.24/ <ATTRIBUTES/> AND <ATTR/>

Tag <attributes/> contains a set of <attr/> . Only one attribute is available in <attributes/> (see table C.28). This attribute activate indicates if the set of attributes is enabled or not.

Name	Type	Read-only	Mandatory	Values
activate	Boolean	no	yes	true, false

TABLE C.28 – <attributes/> Attributes

Tag <attr/> describes named value to pass to spwaned agents. Table C.29 indicates that <attr/> has a name and a value. It may also be activated or not, ie. passed to spawned agents.

C.3.25/ <PROBES/> AND <PROBE/>

Tag <probes/> contains a set of <probe/> . Only one attribute is available in <probes/> (see table C.30). This attribute activate indicates if the set of probes is enabled or not.

Tag <probe/> describes a probe on agents. Table C.31 indicates that <probe/> may be activated or not, and has a name. Probe name is the Java classname of the probe to instance in JASIM.

C.4/ COMPLETE DTD 8.0 FOR 3D SIMULATION

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- $Id$
```

	Name	*Type*	*Read-only*	*Mandatory*	*Values*
activate		Boolean	no	yes	true, false
name		String	no	yes	
value		Variant	no	yes	

TABLE C.29 – <attr/> Attributes

	Name	*Type*	*Read-only*	*Mandatory*	*Values*
activate		Boolean	no	yes	true, false

TABLE C.30 – <attributes/> Attributes

```
    DTD definition for a JaSim configuration file for 3D simulation.
    Version 8.0

    Copyright (c) 2004-10, laboratory Systems and Transport - Computer Science
        Team.
    All rights reserved.

    http://set.utbm.fr/
-->

<!ELEMENT simulation (time?,environment,spawners?)>
<!ATTLIST simulation dtdversion CDATA #FIXED "8.0">
<!ATTLIST simulation id ID #REQUIRED>
<!ATTLIST simulation name CDATA #IMPLIED>
<!ATTLIST simulation date CDATA #IMPLIED>
<!ATTLIST simulation authors CDATA #IMPLIED>
<!ATTLIST simulation version CDATA #IMPLIED>
<!ATTLIST simulation description CDATA #IMPLIED>

<!ELEMENT time EMPTY>
<!ATTLIST time type (step|realtime) "realtime">
<!ATTLIST time unit (millisecond|second|minute|hour) "second">
<!ATTLIST time timeStep CDATA "1">

<!ELEMENT environment (places,portals?)>
<!ATTLIST environment dimension CDATA #FIXED "3d">

<!ELEMENT places (place+)>

<!ELEMENT place (groundEnvironment,staticEnvironment?,dynamicEnvironment?,
    environmentProbes?)>
<!ATTLIST place activate (true|false) "true">
<!ATTLIST place id ID #REQUIRED>
<!ATTLIST place name CDATA #IMPLIED>
<!ATTLIST place perceptionAlgo (sequential|parallel) "sequential">
<!ATTLIST place perceptionTraversal (topdown|bottomup) "topdown">

<!ELEMENT groundEnvironment (heightmap|indoorGround)>
<!ATTLIST groundEnvironment type (alignedIndoor|orientedIndoor|heightmap|
    repulsionHeightmap|any) "any">
```

Name	Type	Read-only	Mandatory	Values
activate	Boolean	no	yes	true, false
name	Classname	no	yes	

TABLE C.31 – <probe/> Attributes

```
<!ATTLIST groundEnvironment id CDATA #REQUIRED>
<!ATTLIST groundEnvironment prebuildResource CDATA #IMPLIED>

<!ELEMENT staticEnvironment EMPTY>
<!ATTLIST staticEnvironment class CDATA #REQUIRED>
<!ATTLIST staticEnvironment prebuildResource CDATA #IMPLIED>

<!ELEMENT dynamicEnvironment (agent*)>
<!ATTLIST dynamicEnvironment class CDATA #IMPLIED>
<!ATTLIST dynamicEnvironment prebuildResource CDATA #IMPLIED>
<!ATTLIST dynamicEnvironment agentType CDATA #IMPLIED>

<!ELEMENT agent (agentSemantic*)>
<!ATTLIST agent activate (true|false) "true">
<!ATTLIST agent class CDATA #REQUIRED>
<!ATTLIST agent type CDATA #REQUIRED>

<!ELEMENT agentSemantics (agentSemantic+)>

<!ELEMENT agentSemantic (#PCDATA)>

<!ELEMENT environmentProbes (environmentProbe+)>
<!ATTLIST environmentProbes activate (true|false) "true">

<!ELEMENT environmentProbe EMPTY>
<!ATTLIST environmentProbe activate (true|false) "true">
<!ATTLIST environmentProbe x CDATA "0">
<!ATTLIST environmentProbe y CDATA "0">
<!ATTLIST environmentProbe z CDATA "0">
<!ATTLIST environmentProbe name ID #REQUIRED>

<!ELEMENT portals (portal+)>
<!ATTLIST portals activate (true|false) "true">

<!ELEMENT portal (portalSource,portalTarget)>
<!ATTLIST portal activate (true|false) "true">

<!ELEMENT portalSource EMPTY>
<!ATTLIST portalSource place IDREF #REQUIRED>
<!ATTLIST portalSource x1 CDATA "0">
<!ATTLIST portalSource y1 CDATA "0">
<!ATTLIST portalSource x2 CDATA "0">
<!ATTLIST portalSource y2 CDATA "0">
<!ATTLIST portalSource onleft (true|false) "true">

<!ELEMENT portalTarget EMPTY>
<!ATTLIST portalTarget place IDREF #REQUIRED>
<!ATTLIST portalTarget x CDATA "0">
```

```
<!ATTLIST portalTarget y CDATA "0">
<!ATTLIST portalTarget dx CDATA "0">
<!ATTLIST portalTarget dy CDATA "0">

<!ELEMENT heightmap EMPTY>
<!ATTLIST heightmap url CDATA #REQUIRED>
<!ATTLIST heightmap minx CDATA #REQUIRED>
<!ATTLIST heightmap miny CDATA #REQUIRED>
<!ATTLIST heightmap minz CDATA #REQUIRED>
<!ATTLIST heightmap maxx CDATA #REQUIRED>
<!ATTLIST heightmap maxy CDATA #REQUIRED>
<!ATTLIST heightmap maxz CDATA #REQUIRED>
<!ATTLIST heightmap rawGroundZero CDATA "-127">
<!ATTLIST heightmap groundZero CDATA #IMPLIED>
<!ATTLIST heightmap semantic CDATA #IMPLIED>

<!ELEMENT indoorGround (alignedArea|orientedArea)>
<!ATTLIST indoorGround z CDATA #REQUIRED>
<!ATTLIST indoorGround semantic CDATA #IMPLIED>

<!ELEMENT alignedArea EMPTY>
<!ATTLIST alignedArea minx CDATA #REQUIRED>
<!ATTLIST alignedArea miny CDATA #REQUIRED>
<!ATTLIST alignedArea maxx CDATA #REQUIRED>
<!ATTLIST alignedArea maxy CDATA #REQUIRED>

<!ELEMENT orientedArea EMPTY>
<!ATTLIST orientedArea centerx CDATA #REQUIRED>
<!ATTLIST orientedArea centery CDATA #REQUIRED>
<!ATTLIST orientedArea Rx CDATA #REQUIRED>
<!ATTLIST orientedArea Ry CDATA #REQUIRED>
<!ATTLIST orientedArea Sx CDATA #REQUIRED>
<!ATTLIST orientedArea Sy CDATA #REQUIRED>

<!ELEMENT spawners (spawner+)>
<!ATTLIST spawners activate (true|false) "true">

<!ELEMENT spawner (entity+)>
<!ATTLIST spawner activate (true|false) "true">
<!ATTLIST spawner type (point|area) "point">
<!ATTLIST spawner id ID #REQUIRED>
<!ATTLIST spawner name CDATA #IMPLIED>
<!ATTLIST spawner place IDREF "">
<!ATTLIST spawner x CDATA "0">
<!ATTLIST spawner y CDATA #IMPLIED>
<!ATTLIST spawner z CDATA #IMPLIED>
<!ATTLIST spawner width CDATA "0">
<!ATTLIST spawner height CDATA "0">
<!ATTLIST spawner startAngle CDATA "0">
<!ATTLIST spawner endAngle CDATA "6.283185308">
<!ATTLIST spawner startDate CDATA "0">
<!ATTLIST spawner endDate CDATA #IMPLIED>

<!ELEMENT entity (frustums,body?,agentSemantics?,waypoints?,goals?,generationLaw,
    attributes?,probes?)>
<!ATTLIST entity activate (true|false) "true">
```

```
<!ATTLIST entity agentType CDATA #REQUIRED>
<!ATTLIST entity budget CDATA #IMPLIED>

<!ELEMENT frustums (frustum+)>

<!ELEMENT frustum EMPTY>
<!ATTLIST frustum type (rectangle|sphere|pyramid|pedestrian) #REQUIRED>
<!ATTLIST frustum eyePosition CDATA #IMPLIED>
<!ATTLIST frustum farDistance CDATA #REQUIRED>
<!ATTLIST frustum sideSize CDATA "0.0">
<!ATTLIST frustum verticalSize CDATA "0.0">
<!ATTLIST frustum nearDistance CDATA "0.0">
<!ATTLIST frustum fieldOfViewHAngle CDATA "0.785398164">
<!ATTLIST frustum fieldOfViewVAngle CDATA "0.785398164">

<!ELEMENT body EMPTY>
<!ATTLIST body type CDATA #IMPLIED>
<!ATTLIST body height CDATA #IMPLIED>
<!ATTLIST body radius CDATA #IMPLIED>

<!ELEMENT waypoints (waypoint+)>
<!ATTLIST waypoints activate (true|false) "true">

<!ELEMENT waypoint EMPTY>
<!ATTLIST waypoint activate (true|false) "true">
<!ATTLIST waypoint name CDATA #IMPLIED>
<!ATTLIST waypoint place IDREF "">
<!ATTLIST waypoint x CDATA "0">
<!ATTLIST waypoint y CDATA "0">
<!ATTLIST waypoint z CDATA "0">
<!ATTLIST waypoint tangentX CDATA #IMPLIED>
<!ATTLIST waypoint tangentY CDATA #IMPLIED>
<!ATTLIST waypoint tangentZ CDATA #IMPLIED>
<!ATTLIST waypoint velocity CDATA #IMPLIED>
<!ATTLIST waypoint time CDATA #IMPLIED>

<!ELEMENT goals (goal+)>
<!ATTLIST goals activate (true|false) "true">

<!ELEMENT goal EMPTY>
<!ATTLIST goal activate (true|false) "true">
<!ATTLIST goal name CDATA #IMPLIED>
<!ATTLIST goal place IDREF "">
<!ATTLIST goal x CDATA "0">
<!ATTLIST goal y CDATA "0">
<!ATTLIST goal z CDATA "0">
<!ATTLIST goal tangentX CDATA #IMPLIED>
<!ATTLIST goal tangentY CDATA #IMPLIED>
<!ATTLIST goal tangentZ CDATA #IMPLIED>
<!ATTLIST goal time CDATA #IMPLIED>

<!ELEMENT generationLaw (lawParam*)>
<!ATTLIST generationLaw class CDATA #REQUIRED>

<!ELEMENT lawParam EMPTY>
<!ATTLIST lawParam name CDATA #REQUIRED>
```

```
<!ATTLIST lawParam value CDATA #REQUIRED>

<!ELEMENT attributes (attr+)>
<!ATTLIST attributes activate (true|false) "true">

<!ELEMENT attr EMPTY>
<!ATTLIST attr activate (true|false) "true">
<!ATTLIST attr name CDATA #REQUIRED>
<!ATTLIST attr value CDATA #REQUIRED>

<!ELEMENT probes (probe+)>
<!ATTLIST probes activate (true|false) "true">

<!ELEMENT probe EMPTY>
<!ATTLIST probe activate (true|false) "true">
<!ATTLIST probe name ID #REQUIRED>
```

C.5/ COMPLETE DTD 7.0 FOR 1.5D SIMULATION

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- $Id$

    DTD definition for a JaSim configuration file for 1.5D simulation.
    Version 7.0

    Copyright (c) 2004-09, laboratory Systems and Transport - Computer Science
        Team.
    All rights reserved.

    http://set.utbm.fr/
-->

<!ELEMENT simulation (time?,environment,spawners?)>
<!ATTLIST simulation dtdversion CDATA #FIXED "7.0">
<!ATTLIST simulation id ID #REQUIRED>
<!ATTLIST simulation name CDATA #IMPLIED>
<!ATTLIST simulation date CDATA #IMPLIED>
<!ATTLIST simulation authors CDATA #IMPLIED>
<!ATTLIST simulation version CDATA #IMPLIED>
<!ATTLIST simulation description CDATA #IMPLIED>

<!ELEMENT time EMPTY>
<!ATTLIST time type (step|realtime) "realtime">
<!ATTLIST time unit (millisecond|second|minute|hour) "second">
<!ATTLIST time timeStep CDATA "1">

<!ELEMENT environment (places,portals?)>
<!ATTLIST environment dimension CDATA #FIXED "1.5d">

<!ELEMENT places (place+)>

<!ELEMENT place (groundEnvironment,staticEnvironment?,dynamicEnvironment?,
    environmentProbes?)>
<!ATTLIST place activate (true|false) "true">
<!ATTLIST place id ID #REQUIRED>
<!ATTLIST place name CDATA #IMPLIED>

<!ELEMENT groundEnvironment EMPTY>
<!ATTLIST groundEnvironment id CDATA #REQUIRED>
<!ATTLIST groundEnvironment shapeFile CDATA #REQUIRED>
<!ATTLIST groundEnvironment dBaseFile CDATA #IMPLIED>

<!ELEMENT staticEnvironment EMPTY>
<!ATTLIST staticEnvironment class CDATA #REQUIRED>
<!ATTLIST staticEnvironment prebuildResource CDATA #IMPLIED>

<!ELEMENT dynamicEnvironment (agent*)>
<!ATTLIST dynamicEnvironment class CDATA #IMPLIED>
<!ATTLIST dynamicEnvironment prebuildResource CDATA #IMPLIED>

<!ELEMENT agent (agentSemantic*)>
```

```
<!ATTLIST agent activate (true|false) "true">
<!ATTLIST agent class CDATA #REQUIRED>
<!ATTLIST agent type CDATA #REQUIRED>

<!ELEMENT agentSemantics (agentSemantic+)>

<!ELEMENT agentSemantic (#PCDATA)>

<!ELEMENT environmentProbes (environmentProbe+)>
<!ATTLIST environmentProbes activate (true|false) "true">

<!ELEMENT environmentProbe EMPTY>
<!ATTLIST environmentProbe activate (true|false) "true">
<!ATTLIST environmentProbe road CDATA #REQUIRED>
<!ATTLIST environmentProbe x CDATA "0">
<!ATTLIST environmentProbe y CDATA "0">
<!ATTLIST environmentProbe name ID #REQUIRED>

<!ELEMENT portals (portal+)>
<!ATTLIST portals activate (true|false) "true">

<!ELEMENT portal (portalSource,portalTarget)>
<!ATTLIST portal activate (true|false) "true">

<!ELEMENT portalSource EMPTY>
<!ATTLIST portalSource place IDREF #REQUIRED>
<!ATTLIST portalSource road CDATA #REQUIRED>
<!ATTLIST portalSource x CDATA "0">
<!ATTLIST portalSource y1 CDATA "0">
<!ATTLIST portalSource y2 CDATA "0">
<!ATTLIST portalSource direction (segment|reverse|both) "both">

<!ELEMENT portalTarget EMPTY>
<!ATTLIST portalTarget place IDREF #REQUIRED>
<!ATTLIST portalTarget road CDATA #REQUIRED>
<!ATTLIST portalTarget x CDATA "0">
<!ATTLIST portalTarget y CDATA "0">
<!ATTLIST portalTarget direction (segment|reverse) "segment">

<!ELEMENT spawners (spawner+)>
<!ATTLIST spawners activate (true|false) "true">

<!ELEMENT spawner (entity+)>
<!ATTLIST spawner activate (true|false) "true">
<!ATTLIST spawner type (point|area) "point">
<!ATTLIST spawner id ID #REQUIRED>
<!ATTLIST spawner name CDATA #IMPLIED>
<!ATTLIST spawner place IDREF "">
<!ATTLIST spawner road CDATA #REQUIRED>
<!ATTLIST spawner x CDATA "0">
<!ATTLIST spawner y CDATA #IMPLIED>
<!ATTLIST spawner width CDATA "0">
<!ATTLIST spawner direction (segment|reverse|both) "both">
<!ATTLIST spawner startDate CDATA "0">
<!ATTLIST spawner endDate CDATA #IMPLIED>
```

```
<!ELEMENT entity (frustums,body?,agentSemantics?,waypoints?,goals?,generationLaw,
    attributes?,probes?)>
<!ATTLIST entity activate (true|false) "true">
<!ATTLIST entity agentType CDATA #REQUIRED>
<!ATTLIST entity budget CDATA #IMPLIED>

<!ELEMENT frustums (frustum+)>

<!ELEMENT frustum EMPTY>
<!ATTLIST frustum type (rectangle) #REQUIRED>
<!ATTLIST frustum farDistance CDATA #REQUIRED>
<!ATTLIST frustum sideSize CDATA #REQUIRED>

<!ELEMENT body EMPTY>
<!ATTLIST body type CDATA #IMPLIED>
<!ATTLIST body curvilineSize CDATA #IMPLIED>
<!ATTLIST body lateralSize CDATA #IMPLIED>

<!ELEMENT waypoints (waypoint+)>
<!ATTLIST waypoints activate (true|false) "true">

<!ELEMENT waypoint EMPTY>
<!ATTLIST waypoint activate (true|false) "true">
<!ATTLIST waypoint name CDATA #IMPLIED>
<!ATTLIST waypoint place IDREF "">
<!ATTLIST waypoint road CDATA #REQUIRED>
<!ATTLIST waypoint x CDATA "0">
<!ATTLIST waypoint y CDATA "0">
<!ATTLIST waypoint direction (segment|reverse|both) "both">
<!ATTLIST waypoint velocity CDATA #IMPLIED>
<!ATTLIST waypoint time CDATA #IMPLIED>

<!ELEMENT goals (goal+)>
<!ATTLIST goals activate (true|false) "true">

<!ELEMENT goal EMPTY>
<!ATTLIST goal activate (true|false) "true">
<!ATTLIST goal name CDATA "">
<!ATTLIST goal place IDREF "">
<!ATTLIST goal road CDATA #REQUIRED>
<!ATTLIST goal x CDATA "0">
<!ATTLIST goal y CDATA "0">
<!ATTLIST goal direction (segment|reverse|both) "both">
<!ATTLIST goal time CDATA #IMPLIED>

<!ELEMENT generationLaw (lawParam*)>
<!ATTLIST generationLaw class CDATA #REQUIRED>

<!ELEMENT lawParam EMPTY>
<!ATTLIST lawParam name CDATA #REQUIRED>
<!ATTLIST lawParam value CDATA #REQUIRED>

<!ELEMENT attributes (attr+)>
<!ATTLIST attributes activate (true|false) "true">

<!ELEMENT attr EMPTY>
```

```
<!ATTLIST attr activate (true|false) "true">
<!ATTLIST attr name CDATA #REQUIRED>
<!ATTLIST attr value CDATA #REQUIRED>

<!ELEMENT probes (probe+)>
<!ATTLIST probes activate (true|false) "true">

<!ELEMENT probe EMPTY>
<!ATTLIST probe activate (true|false) "true">
<!ATTLIST probe name ID #REQUIRED>
```

C.6/ EXAMPLE OF SFG FILE

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE simulation PUBLIC "JaSimConfigurationFile3d v8.0//EN" "jasim-config-3d
    -8.0.dtd">
<simulation id="fb3d4228-4a72-11df-a3f1-0013a928af97"
     date="2010-04-17"
     authors="Jonathan Demange"
     version="0.1"
     description="Simulation of building D of SeT laboratory">

 <time type="step" unit="millisecond" timeStep="200" />

 <environment dimension="3d">
  <places>
  <place id="02bb8d84-4a73-11df-aa89-0013a928af97">
    <groundEnvironment id="0a26b0da-4a73-11df-ba4e-0013a928af97">
    <indoorGround z="4.389826">
      <alignedArea minx="-156.85362368033097"
                   miny="49.7830095944054"
                   maxx="-119.39181399316558"
                   maxy="80.64227430035163" />
    </indoorGround>
    </groundEnvironment>

    <staticEnvironment
    class="jasim.environment.model.perception.tree.structures.
        StaticIcosepQuadTree3D"
    prebuildResource="./jasimFiles/staticWorld.tree" />
  </place>
  </places>
</environment>

<spawners>
 <spawner type="area"
    id="2b067e48-4a73-11df-b4cb-0013a928af97"
    name="Main entry"
    x="-133"
    y="51"
    z="4.389826"
    width="1.8" height="3"
    place="02bb8d84-4a73-11df-aa89-0013a928af97">
 <entity budget="2" agentType="fr.utbm.set.demos.simulation.dbuilding.agent.
     Human" >
```

```
   <frustums>
   <frustum type="sphere"
      eyePosition="1.8"
      farDistance="8" />
   </frustums>

   <body type="fr.utbm.set.jasim.environment.semantics.PedestrianType"
        height="1.8"
        radius="0.6" />

   <waypoints>
   <waypoint name="Office_Demange"
       place="02bb8d84-4a73-11df-aa89-0013a928af97"
       x="-140"
       y="78"
       z="4.389826" />
   </waypoints>

   <goals>
   <goal name="Goal1"
        place="02bb8d84-4a73-11df-aa89-0013a928af97"
        x="-140"
        y="70"
        z="4.389826" />
   </goals>

   <generationLaw class="fr.utbm.set.jasim.spawn.ConstantSpawningLaw">
   <lawParam name="value" value= "10000"/>
   </generationLaw>
  </entity>
  </spawner>
 </spawners>
</simulation>
```

C.7/ CHANGES IN DTD FOR 3D SIMULATION

This section describes the changes from a DTD version to another DTD version.

C.7.1/ 7.0 → 8.0

This section describes all the changes to apply too pass from DTD version 7.0 to DTD version 8.0.

C.7.1.1/ ATTRIBUTE ADDITIONS

Table C.32 describes all the new attributes in DTD 8.0. This table also shows the value of the attributes in DTD 8.0 which is corresponding to the default behavior in 7.0.

Element	*Attribute*	*Value assumed in 7.0*
place	perceptionAlgo	sequential

TABLE C.32 – New Attributes in DTD 8.0

C.7.1.2/ Attribute Translations

Table C.33 describes all the translation of attributes from DTD 7.0 to DTD 8.0. This table gives the values in DTD 7.0 and the corresponding values in 8.0.

Element	*Attribute*	*Value in 7.0*	*Value in 8.0*
groundEnvironment	type	constant	alignedIndoor, orientedIndoor
groundEnvironment	type	simple	heightmap
groundEnvironment	type	repulsion	repulsionHeightmap

TABLE C.33 – Translate Attributes from DTD 7.0 to 8.0

C.7.1.3/ Node Additions

Table C.34 describes all the new nodes in DTD 8.0. This table also shows the node in DTD 8.0 which is corresponding to the default behavior in 7.0.

Element	*New Child Node*	*Node assumed in 7.0*
indoorGround	alignedArea, orientedArea	alignedArea

TABLE C.34 – New Nodes in DTD 8.0

D

CONFIGURATION INITIALE DU SCÉNARIO DE SIMULATION D'UN AÉROPORT

Cette annexe présente le fichier de configuration pour la simulation d'un aéroport utilisé dans le chapitre 6. Ce fichier de configuration est un fichier XML dont les spécifications sont décrites dans l'annexe C.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE simulation PUBLIC "-//set.utbm.fr//DTD JaSimConfigurationFile3dv7.0//EN
    " "/fr/utbm/set/jasim/controller/config/jasim-config-3d-7.0.dtd">
<simulation id="2284f02e-50d8-40fc-9cbe-f8bcce81249d" name="Ped-Toy-Demo-2011"
  date="2011-10-28" authors="DEMANGE Jonathan" version="2.3"
  description="Pedestrian Simulation">

<time type="step" unit="millisecond" timeStep="500" />

<environment dimension="3d">
  <places>
  <place id="05bde813-aeca-472c-8395-ce4bbbb8d47f" name="entry-hall">
    <groundEnvironment id="6cff72c9-133f-4e36-9ac4-2ca0beebcdfe" type="repulsion"
        >
    <heightmap url="/fr/utbm/set/demo/simulation/heightmap/ground_airport_entry.
        png"
        minx="0"
        miny="0"
        minz="0"
        maxx="1800"
        maxy="735"
        maxz="0"
        rawGroundZero="-127"
        semantic="fr.utbm.set.jasim.environment.semantics.GroundType" />
    </groundEnvironment>

    <dynamicEnvironment class="fr.utbm.set.jasim.environment.model.perception.
        tree.structures.quadtree.DynamicIcosepQuadTreeMultiLvl3D"
           prebuildResource=""
           agentType="">
      <agent class="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
            type="fr.utbm.set.jasim.environment.semantics.PedestrianType" />
    </dynamicEnvironment>
  </place>
  <place id="c46a25d0-8663-11e0-9d78-0800200c9a66" name="board-hall-one">
```

```
    <groundEnvironment id="5262e6d0-8694-11e0-9d78-0800200c9a66" type="repulsion"
        >
    <heightmap url="/fr/utbm/set/demo/simulation/heightmap/ground_airport_board1.
        png"
         minx="0"
         miny="735"
         minz="0"
         maxx="715"
         maxy="1200"
         maxz="0"
         rawGroundZero="-127"
         semantic="fr.utbm.set.jasim.environment.semantics.GroundType" />
    </groundEnvironment>

    <dynamicEnvironment class="fr.utbm.set.jasim.environment.model.perception.
        tree.structures.quadtree.DynamicIcosepQuadTreeMultiLvl3D"
           prebuildResource=""
           agentType="">
      <agent class="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
            type="fr.utbm.set.jasim.environment.semantics.PedestrianType" />
    </dynamicEnvironment>
  </place>
  <place id="ef98aec0-8663-11e0-9d78-0800200c9a66" name="board-hall-two">
    <groundEnvironment id="5f5d0b90-8694-11e0-9d78-0800200c9a66" type="repulsion"
        >
    <heightmap url="/fr/utbm/set/demo/simulation/heightmap/ground_airport_board2.
        png"
         minx="715"
         miny="735"
         minz="0"
         maxx="1800"
         maxy="1200"
         maxz="0"
         rawGroundZero="-127"
         semantic="fr.utbm.set.jasim.environment.semantics.GroundType" />
    </groundEnvironment>

    <dynamicEnvironment class="fr.utbm.set.jasim.environment.model.perception.
        tree.structures.quadtree.DynamicIcosepQuadTreeMultiLvl3D"
           prebuildResource=""
           agentType="">
      <agent class="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
            type="fr.utbm.set.jasim.environment.semantics.PedestrianType" />
    </dynamicEnvironment>
  </place>
  </places>
</environment>


<spawners>
<!-- Spawners in entry hall after initialisation -->
<spawner type="area" id="116b5fb4-432a-4948-ac80-419d6855ffb0" name="RUN1"
  x="620" y="150" z="0"
  width="50" height="85"
  startAngle="0" endAngle="6.283185308"
  startDate="0" endDate=""
```

```
  place="05bde813-aeca-472c-8395-ce4bbbb8d47f">
  <entity budget="250"
    agentType="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
    type="fr.utbm.set.jasim.environment.semantics.PedestrianType">
    <frustums>
    <frustum type="pedestrian" eyePosition="1.8" farDistance="100"
      nearDistance="10" fieldOfViewHAngle="0.785398164"
      fieldOfViewVAngle="0.785398164"/>
    </frustums>
    <body type="" height="" radius="" />
    <generationLaw class="fr.utbm.set.jasim.spawn.ConstantSpawningLaw">
      <lawParam name="value" value="500" />
    </generationLaw>
    <goals>
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="479" y="470" z="0" tangentX="1" tangentY="0" tangentZ="0" />
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="1232.5" y="660" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    <goal activate="true" place="ef98aec0-8663-11e0-9d78-0800200c9a66"
      x="1320" y="1180" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    </goals>
  </entity>
</spawner>
<spawner type="area" id="116b5fb4-432a-4948-ac80-419d6855ffb1" name="RUN2"
  x="620" y="300" z="0"
  width="50" height="85"
  startAngle="0" endAngle="6.283185308"
  startDate="0" endDate=""
  place="05bde813-aeca-472c-8395-ce4bbbb8d47f">
  <entity budget="250"
    agentType="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
    type="fr.utbm.set.jasim.environment.semantics.PedestrianType">
    <frustums>
    <frustum type="pedestrian" eyePosition="1.8" farDistance="100"
      nearDistance="10" fieldOfViewHAngle="0.785398164"
      fieldOfViewVAngle="0.785398164"/>
    </frustums>
    <body type="" height="" radius="" />
    <generationLaw class="fr.utbm.set.jasim.spawn.ConstantSpawningLaw">
      <lawParam name="value" value="500" />
    </generationLaw>
    <goals>
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="479" y="310" z="0" tangentX="1" tangentY="0" tangentZ="0" />
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="567.5" y="660" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    <goal activate="true" place="c46a25d0-8663-11e0-9d78-0800200c9a66"
      x="540" y="1180" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    </goals>
  </entity>
</spawner>
<spawner type="area" id="116b5fb4-432a-4948-ac80-419d6855ffb2" name="RUN3"
  x="620" y="450" z="0"
  width="50" height="85"
  startAngle="0" endAngle="6.283185308"
  startDate="0" endDate=""
```

```
  place="05bde813-aeca-472c-8395-ce4bbbb8d47f">
  <entity budget="250"
    agentType="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
    type="fr.utbm.set.jasim.environment.semantics.PedestrianType">
    <frustums>
    <frustum type="pedestrian" eyePosition="1.8" farDistance="100"
      nearDistance="10" fieldOfViewHAngle="0.785398164"
      fieldOfViewVAngle="0.785398164"/>
    </frustums>
    <body type="" height="" radius="" />
    <generationLaw class="fr.utbm.set.jasim.spawn.ConstantSpawningLaw">
      <lawParam name="value" value="500" />
    </generationLaw>
    <goals>
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="479" y="310" z="0" tangentX="1" tangentY="0" tangentZ="0" />
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="567.5" y="660" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    <goal activate="true" place="c46a25d0-8663-11e0-9d78-0800200c9a66"
      x="540" y="1180" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    </goals>
  </entity>
</spawner>
<spawner type="area" id="116b5fb4-432a-4948-ac80-419d6855ffb3" name="RUN4"
  x="1130" y="150" z="0"
  width="50" height="85"
  startAngle="0" endAngle="6.283185308"
  startDate="0" endDate=""
  place="05bde813-aeca-472c-8395-ce4bbbb8d47f">
  <entity budget="250"
    agentType="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
    type="fr.utbm.set.jasim.environment.semantics.PedestrianType">
    <frustums>
    <frustum type="pedestrian" eyePosition="1.8" farDistance="100"
      nearDistance="10" fieldOfViewHAngle="0.785398164"
      fieldOfViewVAngle="0.785398164"/>
    </frustums>
    <body type="" height="" radius="" />
    <generationLaw class="fr.utbm.set.jasim.spawn.ConstantSpawningLaw">
      <lawParam name="value" value="500" />
    </generationLaw>
    <goals>
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="1325" y="450" z="0" tangentX="1" tangentY="0" tangentZ="0" />
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="1232.5" y="660" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    <goal activate="true" place="ef98aec0-8663-11e0-9d78-0800200c9a66"
      x="1320" y="1180" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    </goals>
  </entity>
</spawner>
<spawner type="area" id="116b5fb4-432a-4948-ac80-419d6855ffb4" name="RUN5"
  x="1130" y="300" z="0"
  width="50" height="85"
  startAngle="0" endAngle="6.283185308"
  startDate="0" endDate=""
```

```
  place="05bde813-aeca-472c-8395-ce4bbbb8d47f">
  <entity budget="250"
    agentType="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
    type="fr.utbm.set.jasim.environment.semantics.PedestrianType">
    <frustums>
    <frustum type="pedestrian" eyePosition="1.8" farDistance="100"
      nearDistance="10" fieldOfViewHAngle="0.785398164"
      fieldOfViewVAngle="0.785398164"/>
    </frustums>
    <body type="" height="" radius="" />
    <generationLaw class="fr.utbm.set.jasim.spawn.ConstantSpawningLaw">
      <lawParam name="value" value="500" />
    </generationLaw>
    <goals>
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="1325" y="190" z="0" tangentX="1" tangentY="0" tangentZ="0" />
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="1232.5" y="660" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    <goal activate="true" place="ef98aec0-8663-11e0-9d78-0800200c9a66"
      x="1320" y="1180" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    </goals>
  </entity>
</spawner>
<spawner type="area" id="116b5fb4-432a-4948-ac80-419d6855ffb5" name="RUN6"
  x="1130" y="450" z="0"
  width="50" height="85"
  startAngle="0" endAngle="6.283185308"
  startDate="0" endDate=""
  place="05bde813-aeca-472c-8395-ce4bbbb8d47f">
  <entity budget="250"
    agentType="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
    type="fr.utbm.set.jasim.environment.semantics.PedestrianType">
    <frustums>
    <frustum type="pedestrian" eyePosition="1.8" farDistance="100"
      nearDistance="10" fieldOfViewHAngle="0.785398164"
      fieldOfViewVAngle="0.785398164"/>
    </frustums>
    <body type="" height="" radius="" />
    <generationLaw class="fr.utbm.set.jasim.spawn.ConstantSpawningLaw">
      <lawParam name="value" value="500" />
    </generationLaw>
    <goals>
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="1325" y="190" z="0" tangentX="1" tangentY="0" tangentZ="0" />
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="1232.5" y="660" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    <goal activate="true" place="ef98aec0-8663-11e0-9d78-0800200c9a66"
      x="1320" y="1180" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    </goals>
  </entity>
</spawner>
<spawner type="area" id="116b5fb4-432a-4948-ac80-419d6855ffb6" name="RUN7"
  x="490" y="150" z="0"
  width="50" height="85"
  startAngle="0" endAngle="6.283185308"
  startDate="0" endDate=""
```

```
  place="05bde813-aeca-472c-8395-ce4bbbb8d47f">
  <entity budget="250"
    agentType="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
    type="fr.utbm.set.jasim.environment.semantics.PedestrianType">
    <frustums>
    <frustum type="pedestrian" eyePosition="1.8" farDistance="100"
      nearDistance="10" fieldOfViewHAngle="0.785398164"
      fieldOfViewVAngle="0.785398164"/>
    </frustums>
    <body type="" height="" radius="" />
    <generationLaw class="fr.utbm.set.jasim.spawn.ConstantSpawningLaw">
      <lawParam name="value" value="500" />
    </generationLaw>
    <goals>
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="479" y="470" z="0" tangentX="1" tangentY="0" tangentZ="0" />
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="1232.5" y="660" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    <goal activate="true" place="ef98aec0-8663-11e0-9d78-0800200c9a66"
      x="1320" y="1180" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    </goals>
  </entity>
</spawner>
<spawner type="area" id="116b5fb4-432a-4948-ac80-419d6855ffb7" name="RUN8"
  x="490" y="150" z="0"
  width="50" height="85"
  startAngle="0" endAngle="6.283185308"
  startDate="0" endDate=""
  place="05bde813-aeca-472c-8395-ce4bbbb8d47f">
  <entity budget="250"
    agentType="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
    type="fr.utbm.set.jasim.environment.semantics.PedestrianType">
    <frustums>
    <frustum type="pedestrian" eyePosition="1.8" farDistance="100"
      nearDistance="10" fieldOfViewHAngle="0.785398164"
      fieldOfViewVAngle="0.785398164"/>
    </frustums>
    <body type="" height="" radius="" />
    <generationLaw class="fr.utbm.set.jasim.spawn.ConstantSpawningLaw">
      <lawParam name="value" value="500" />
    </generationLaw>
    <goals>
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="479" y="310" z="0" tangentX="1" tangentY="0" tangentZ="0" />
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="567.5" y="660" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    <goal activate="true" place="c46a25d0-8663-11e0-9d78-0800200c9a66"
      x="540" y="1180" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    </goals>
  </entity>
</spawner>
<spawner type="area" id="116b5fb4-432a-4948-ac80-419d6855ffb8" name="RUN9"
  x="490" y="450" z="0"
  width="50" height="85"
  startAngle="0" endAngle="6.283185308"
  startDate="0" endDate=""
```

```
  place="05bde813-aeca-472c-8395-ce4bbbb8d47f">
  <entity budget="250"
    agentType="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
    type="fr.utbm.set.jasim.environment.semantics.PedestrianType">
    <frustums>
    <frustum type="pedestrian" eyePosition="1.8" farDistance="100"
      nearDistance="10" fieldOfViewHAngle="0.785398164"
      fieldOfViewVAngle="0.785398164"/>
    </frustums>
    <body type="" height="" radius="" />
    <generationLaw class="fr.utbm.set.jasim.spawn.ConstantSpawningLaw">
      <lawParam name="value" value="500" />
    </generationLaw>
    <goals>
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="479" y="310" z="0" tangentX="1" tangentY="0" tangentZ="0" />
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="567.5" y="660" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    <goal activate="true" place="c46a25d0-8663-11e0-9d78-0800200c9a66"
      x="540" y="1180" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    </goals>
  </entity>
</spawner>
<spawner type="area" id="116b5fb4-432a-4948-ac80-419d6855ffb9" name="RUN10"
  x="1280" y="150" z="0"
  width="50" height="85"
  startAngle="0" endAngle="6.283185308"
  startDate="0" endDate=""
  place="05bde813-aeca-472c-8395-ce4bbbb8d47f">
  <entity budget="250"
    agentType="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
    type="fr.utbm.set.jasim.environment.semantics.PedestrianType">
    <frustums>
    <frustum type="pedestrian" eyePosition="1.8" farDistance="100"
      nearDistance="10" fieldOfViewHAngle="0.785398164"
      fieldOfViewVAngle="0.785398164"/>
    </frustums>
    <body type="" height="" radius="" />
    <generationLaw class="fr.utbm.set.jasim.spawn.ConstantSpawningLaw">
      <lawParam name="value" value="500" />
    </generationLaw>
    <goals>
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="1325" y="450" z="0" tangentX="1" tangentY="0" tangentZ="0" />
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="1232.5" y="660" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    <goal activate="true" place="ef98aec0-8663-11e0-9d78-0800200c9a66"
      x="1320" y="1180" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    </goals>
  </entity>
</spawner>
<spawner type="area" id="116b5fb4-432a-4948-ac80-419d6855ffc0" name="RUN11"
  x="1280" y="300" z="0"
  width="50" height="85"
  startAngle="0" endAngle="6.283185308"
  startDate="0" endDate=""
```

```
  place="05bde813-aeca-472c-8395-ce4bbbb8d47f">
  <entity budget="250"
    agentType="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
    type="fr.utbm.set.jasim.environment.semantics.PedestrianType">
    <frustums>
    <frustum type="pedestrian" eyePosition="1.8" farDistance="100"
      nearDistance="10" fieldOfViewHAngle="0.785398164"
      fieldOfViewVAngle="0.785398164"/>
    </frustums>
    <body type="" height="" radius="" />
    <generationLaw class="fr.utbm.set.jasim.spawn.ConstantSpawningLaw">
      <lawParam name="value" value="500" />
    </generationLaw>
    <goals>
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="1325" y="190" z="0" tangentX="1" tangentY="0" tangentZ="0" />
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="1232.5" y="660" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    <goal activate="true" place="ef98aec0-8663-11e0-9d78-0800200c9a66"
      x="1320" y="1180" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    </goals>
  </entity>
</spawner>
<spawner type="area" id="116b5fb4-432a-4948-ac80-419d6855ffc1" name="RUN12"
  x="1280" y="450" z="0"
  width="50" height="85"
  startAngle="0" endAngle="6.283185308"
  startDate="0" endDate=""
  place="05bde813-aeca-472c-8395-ce4bbbb8d47f">
  <entity budget="250"
    agentType="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
    type="fr.utbm.set.jasim.environment.semantics.PedestrianType">
    <frustums>
    <frustum type="pedestrian" eyePosition="1.8" farDistance="100"
      nearDistance="10" fieldOfViewHAngle="0.785398164"
      fieldOfViewVAngle="0.785398164"/>
    </frustums>
    <body type="" height="" radius="" />
    <generationLaw class="fr.utbm.set.jasim.spawn.ConstantSpawningLaw">
      <lawParam name="value" value="500" />
    </generationLaw>
    <goals>
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="1325" y="190" z="0" tangentX="1" tangentY="0" tangentZ="0" />
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="1232.5" y="660" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    <goal activate="true" place="ef98aec0-8663-11e0-9d78-0800200c9a66"
      x="1320" y="1180" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    </goals>
  </entity>
</spawner>


<!-- Spawners in entry hall for initialisation -->
<spawner type="area" id="116b5fb4-432a-4948-ac80-419d6855ffb0" name="START1"
  x="620" y="150" z="0"
```

```
  width="50" height="85"
  startAngle="0" endAngle="6.283185308"
  startDate="0" endDate=""
  place="05bde813-aeca-472c-8395-ce4bbbb8d47f">
  <entity budget="50"
    agentType="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
    type="fr.utbm.set.jasim.environment.semantics.PedestrianType">
    <frustums>
    <frustum type="pedestrian" eyePosition="1.8" farDistance="100"
      nearDistance="10" fieldOfViewHAngle="0.785398164"
      fieldOfViewVAngle="0.785398164"/>
    </frustums>
    <body type="" height="" radius="" />
    <generationLaw class="fr.utbm.set.jasim.spawn.ConstantSpawningLaw">
      <lawParam name="value" value="1000000000000" />
    </generationLaw>
    <goals>
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="479" y="470" z="0" tangentX="1" tangentY="0" tangentZ="0" />
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="1232.5" y="660" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    <goal activate="true" place="ef98aec0-8663-11e0-9d78-0800200c9a66"
      x="1320" y="1180" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    </goals>
  </entity>
</spawner>
<spawner type="area" id="116b5fb4-432a-4948-ac80-419d6855ffb1" name="START2"
  x="620" y="300" z="0"
  width="50" height="85"
  startAngle="0" endAngle="6.283185308"
  startDate="0" endDate=""
  place="05bde813-aeca-472c-8395-ce4bbbb8d47f">
  <entity budget="50"
    agentType="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
    type="fr.utbm.set.jasim.environment.semantics.PedestrianType">
    <frustums>
    <frustum type="pedestrian" eyePosition="1.8" farDistance="100"
      nearDistance="10" fieldOfViewHAngle="0.785398164"
      fieldOfViewVAngle="0.785398164"/>
    </frustums>
    <body type="" height="" radius="" />
    <generationLaw class="fr.utbm.set.jasim.spawn.ConstantSpawningLaw">
      <lawParam name="value" value="1000000000000" />
    </generationLaw>
    <goals>
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="479" y="310" z="0" tangentX="1" tangentY="0" tangentZ="0" />
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="567.5" y="660" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    <goal activate="true" place="c46a25d0-8663-11e0-9d78-0800200c9a66"
      x="540" y="1180" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    </goals>
  </entity>
</spawner>
<spawner type="area" id="116b5fb4-432a-4948-ac80-419d6855ffb2" name="START3"
  x="620" y="450" z="0"
```

```
  width="50" height="85"
  startAngle="0" endAngle="6.283185308"
  startDate="0" endDate=""
  place="05bde813-aeca-472c-8395-ce4bbbb8d47f">
  <entity budget="50"
    agentType="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
    type="fr.utbm.set.jasim.environment.semantics.PedestrianType">
    <frustums>
    <frustum type="pedestrian" eyePosition="1.8" farDistance="100"
      nearDistance="10" fieldOfViewHAngle="0.785398164"
      fieldOfViewVAngle="0.785398164"/>
    </frustums>
    <body type="" height="" radius="" />
    <generationLaw class="fr.utbm.set.jasim.spawn.ConstantSpawningLaw">
      <lawParam name="value" value="1000000000000" />
    </generationLaw>
    <goals>
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="479" y="310" z="0" tangentX="1" tangentY="0" tangentZ="0" />
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="567.5" y="660" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    <goal activate="true" place="c46a25d0-8663-11e0-9d78-0800200c9a66"
      x="540" y="1180" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    </goals>
  </entity>
</spawner>
<spawner type="area" id="116b5fb4-432a-4948-ac80-419d6855ffb3" name="START4"
  x="1130" y="150" z="0"
  width="50" height="85"
  startAngle="0" endAngle="6.283185308"
  startDate="0" endDate=""
  place="05bde813-aeca-472c-8395-ce4bbbb8d47f">
  <entity budget="50"
    agentType="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
    type="fr.utbm.set.jasim.environment.semantics.PedestrianType">
    <frustums>
    <frustum type="pedestrian" eyePosition="1.8" farDistance="100"
      nearDistance="10" fieldOfViewHAngle="0.785398164"
      fieldOfViewVAngle="0.785398164"/>
    </frustums>
    <body type="" height="" radius="" />
    <generationLaw class="fr.utbm.set.jasim.spawn.ConstantSpawningLaw">
      <lawParam name="value" value="1000000000000" />
    </generationLaw>
    <goals>
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="1325" y="450" z="0" tangentX="1" tangentY="0" tangentZ="0" />
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="1232.5" y="660" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    <goal activate="true" place="ef98aec0-8663-11e0-9d78-0800200c9a66"
      x="1320" y="1180" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    </goals>
  </entity>
</spawner>
<spawner type="area" id="116b5fb4-432a-4948-ac80-419d6855ffb4" name="START5"
  x="1130" y="300" z="0"
```

```
  width="50" height="85"
  startAngle="0" endAngle="6.283185308"
  startDate="0" endDate=""
  place="05bde813-aeca-472c-8395-ce4bbbb8d47f">
  <entity budget="50"
    agentType="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
    type="fr.utbm.set.jasim.environment.semantics.PedestrianType">
    <frustums>
    <frustum type="pedestrian" eyePosition="1.8" farDistance="100"
      nearDistance="10" fieldOfViewHAngle="0.785398164"
      fieldOfViewVAngle="0.785398164"/>
    </frustums>
    <body type="" height="" radius="" />
    <generationLaw class="fr.utbm.set.jasim.spawn.ConstantSpawningLaw">
      <lawParam name="value" value="1000000000000" />
    </generationLaw>
    <goals>
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="1325" y="190" z="0" tangentX="1" tangentY="0" tangentZ="0" />
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="1232.5" y="660" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    <goal activate="true" place="ef98aec0-8663-11e0-9d78-0800200c9a66"
      x="1320" y="1180" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    </goals>
  </entity>
</spawner>
<spawner type="area" id="116b5fb4-432a-4948-ac80-419d6855ffb5" name="START6"
  x="1130" y="450" z="0"
  width="50" height="85"
  startAngle="0" endAngle="6.283185308"
  startDate="0" endDate=""
  place="05bde813-aeca-472c-8395-ce4bbbb8d47f">
  <entity budget="50"
    agentType="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
    type="fr.utbm.set.jasim.environment.semantics.PedestrianType">
    <frustums>
    <frustum type="pedestrian" eyePosition="1.8" farDistance="100"
      nearDistance="10" fieldOfViewHAngle="0.785398164"
      fieldOfViewVAngle="0.785398164"/>
    </frustums>
    <body type="" height="" radius="" />
    <generationLaw class="fr.utbm.set.jasim.spawn.ConstantSpawningLaw">
      <lawParam name="value" value="1000000000000" />
    </generationLaw>
    <goals>
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="1325" y="190" z="0" tangentX="1" tangentY="0" tangentZ="0" />
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="1232.5" y="660" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    <goal activate="true" place="ef98aec0-8663-11e0-9d78-0800200c9a66"
      x="1320" y="1180" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    </goals>
  </entity>
</spawner>
<spawner type="area" id="116b5fb4-432a-4948-ac80-419d6855ffb6" name="START7"
  x="490" y="150" z="0"
```

```
  width="50" height="85"
  startAngle="0" endAngle="6.283185308"
  startDate="0" endDate=""
  place="05bde813-aeca-472c-8395-ce4bbbb8d47f">
  <entity budget="50"
    agentType="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
    type="fr.utbm.set.jasim.environment.semantics.PedestrianType">
    <frustums>
    <frustum type="pedestrian" eyePosition="1.8" farDistance="100"
      nearDistance="10" fieldOfViewHAngle="0.785398164"
      fieldOfViewVAngle="0.785398164"/>
    </frustums>
    <body type="" height="" radius="" />
    <generationLaw class="fr.utbm.set.jasim.spawn.ConstantSpawningLaw">
      <lawParam name="value" value="1000000000000" />
    </generationLaw>
    <goals>
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="479" y="470" z="0" tangentX="1" tangentY="0" tangentZ="0" />
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="1232.5" y="660" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    <goal activate="true" place="ef98aec0-8663-11e0-9d78-0800200c9a66"
      x="1320" y="1180" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    </goals>
  </entity>
</spawner>
<spawner type="area" id="116b5fb4-432a-4948-ac80-419d6855ffb7" name="START8"
  x="490" y="150" z="0"
  width="50" height="85"
  startAngle="0" endAngle="6.283185308"
  startDate="0" endDate=""
  place="05bde813-aeca-472c-8395-ce4bbbb8d47f">
  <entity budget="50"
    agentType="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
    type="fr.utbm.set.jasim.environment.semantics.PedestrianType">
    <frustums>
    <frustum type="pedestrian" eyePosition="1.8" farDistance="100"
      nearDistance="10" fieldOfViewHAngle="0.785398164"
      fieldOfViewVAngle="0.785398164"/>
    </frustums>
    <body type="" height="" radius="" />
    <generationLaw class="fr.utbm.set.jasim.spawn.ConstantSpawningLaw">
      <lawParam name="value" value="1000000000000" />
    </generationLaw>
    <goals>
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="479" y="310" z="0" tangentX="1" tangentY="0" tangentZ="0" />
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="567.5" y="660" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    <goal activate="true" place="c46a25d0-8663-11e0-9d78-0800200c9a66"
      x="540" y="1180" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    </goals>
  </entity>
</spawner>
<spawner type="area" id="116b5fb4-432a-4948-ac80-419d6855ffb8" name="START9"
  x="490" y="450" z="0"
```

```
  width="50" height="85"
  startAngle="0" endAngle="6.283185308"
  startDate="0" endDate=""
  place="05bde813-aeca-472c-8395-ce4bbbb8d47f">
  <entity budget="50"
    agentType="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
    type="fr.utbm.set.jasim.environment.semantics.PedestrianType">
    <frustums>
    <frustum type="pedestrian" eyePosition="1.8" farDistance="100"
      nearDistance="10" fieldOfViewHAngle="0.785398164"
      fieldOfViewVAngle="0.785398164"/>
    </frustums>
    <body type="" height="" radius="" />
    <generationLaw class="fr.utbm.set.jasim.spawn.ConstantSpawningLaw">
      <lawParam name="value" value="1000000000000" />
    </generationLaw>
    <goals>
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="479" y="310" z="0" tangentX="1" tangentY="0" tangentZ="0" />
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="567.5" y="660" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    <goal activate="true" place="c46a25d0-8663-11e0-9d78-0800200c9a66"
      x="540" y="1180" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    </goals>
  </entity>
</spawner>
<spawner type="area" id="116b5fb4-432a-4948-ac80-419d6855ffb9" name="START10"
  x="1280" y="150" z="0"
  width="50" height="85"
  startAngle="0" endAngle="6.283185308"
  startDate="0" endDate=""
  place="05bde813-aeca-472c-8395-ce4bbbb8d47f">
  <entity budget="50"
    agentType="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
    type="fr.utbm.set.jasim.environment.semantics.PedestrianType">
    <frustums>
    <frustum type="pedestrian" eyePosition="1.8" farDistance="100"
      nearDistance="10" fieldOfViewHAngle="0.785398164"
      fieldOfViewVAngle="0.785398164"/>
    </frustums>
    <body type="" height="" radius="" />
    <generationLaw class="fr.utbm.set.jasim.spawn.ConstantSpawningLaw">
      <lawParam name="value" value="1000000000000" />
    </generationLaw>
    <goals>
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="1325" y="450" z="0" tangentX="1" tangentY="0" tangentZ="0" />
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="1232.5" y="660" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    <goal activate="true" place="ef98aec0-8663-11e0-9d78-0800200c9a66"
      x="1320" y="1180" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    </goals>
  </entity>
</spawner>
<spawner type="area" id="116b5fb4-432a-4948-ac80-419d6855ffc0" name="START11"
  x="1280" y="300" z="0"
```

```
  width="50" height="85"
  startAngle="0" endAngle="6.283185308"
  startDate="0" endDate=""
  place="05bde813-aeca-472c-8395-ce4bbbb8d47f">
  <entity budget="50"
    agentType="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
    type="fr.utbm.set.jasim.environment.semantics.PedestrianType">
    <frustums>
    <frustum type="pedestrian" eyePosition="1.8" farDistance="100"
      nearDistance="10" fieldOfViewHAngle="0.785398164"
      fieldOfViewVAngle="0.785398164"/>
    </frustums>
    <body type="" height="" radius="" />
    <generationLaw class="fr.utbm.set.jasim.spawn.ConstantSpawningLaw">
      <lawParam name="value" value="1000000000000" />
    </generationLaw>
    <goals>
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="1325" y="190" z="0" tangentX="1" tangentY="0" tangentZ="0" />
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="1232.5" y="660" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    <goal activate="true" place="ef98aec0-8663-11e0-9d78-0800200c9a66"
      x="1320" y="1180" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    </goals>
  </entity>
</spawner>
<spawner type="area" id="116b5fb4-432a-4948-ac80-419d6855ffc1" name="START12"
  x="1280" y="450" z="0"
  width="50" height="85"
  startAngle="0" endAngle="6.283185308"
  startDate="0" endDate=""
  place="05bde813-aeca-472c-8395-ce4bbbb8d47f">
  <entity budget="50"
    agentType="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
    type="fr.utbm.set.jasim.environment.semantics.PedestrianType">
    <frustums>
    <frustum type="pedestrian" eyePosition="1.8" farDistance="100"
      nearDistance="10" fieldOfViewHAngle="0.785398164"
      fieldOfViewVAngle="0.785398164"/>
    </frustums>
    <body type="" height="" radius="" />
    <generationLaw class="fr.utbm.set.jasim.spawn.ConstantSpawningLaw">
      <lawParam name="value" value="1000000000000" />
    </generationLaw>
    <goals>
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="1325" y="190" z="0" tangentX="1" tangentY="0" tangentZ="0" />
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="1232.5" y="660" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    <goal activate="true" place="ef98aec0-8663-11e0-9d78-0800200c9a66"
      x="1320" y="1180" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    </goals>
  </entity>
</spawner>
<spawner type="area" id="116b5fb4-432a-4948-ac80-419d6855ffc2" name="START13"
  x="700" y="400" z="0"
```

```
  width="400" height="150"
  startAngle="0" endAngle="6.283185308"
  startDate="0" endDate=""
  place="05bde813-aeca-472c-8395-ce4bbbb8d47f">
  <entity budget="40"
    agentType="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
    type="fr.utbm.set.jasim.environment.semantics.PedestrianType">
    <frustums>
    <frustum type="pedestrian" eyePosition="1.8" farDistance="100"
      nearDistance="10" fieldOfViewHAngle="0.785398164"
      fieldOfViewVAngle="0.785398164"/>
    </frustums>
    <body type="" height="" radius="" />
    <generationLaw class="fr.utbm.set.jasim.spawn.ConstantSpawningLaw">
      <lawParam name="value" value="1000000000000" />
    </generationLaw>
    <goals>
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="1232.5" y="660" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    <goal activate="true" place="ef98aec0-8663-11e0-9d78-0800200c9a66"
      x="1320" y="1180" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    </goals>
  </entity>
</spawner>
<spawner type="area" id="116b5fb4-432a-4948-ac80-419d6855ffc3" name="START14"
  x="700" y="400" z="0"
  width="400" height="150"
  startAngle="0" endAngle="6.283185308"
  startDate="0" endDate=""
  place="05bde813-aeca-472c-8395-ce4bbbb8d47f">
  <entity budget="40"
    agentType="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
    type="fr.utbm.set.jasim.environment.semantics.PedestrianType">
    <frustums>
    <frustum type="pedestrian" eyePosition="1.8" farDistance="100"
      nearDistance="10" fieldOfViewHAngle="0.785398164"
      fieldOfViewVAngle="0.785398164"/>
    </frustums>
    <body type="" height="" radius="" />
    <generationLaw class="fr.utbm.set.jasim.spawn.ConstantSpawningLaw">
      <lawParam name="value" value="1000000000000" />
    </generationLaw>
    <goals>
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="567.5" y="660" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    <goal activate="true" place="c46a25d0-8663-11e0-9d78-0800200c9a66"
      x="540" y="1180" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    </goals>
  </entity>
</spawner>
<spawner type="area" id="116b5fb4-432a-4948-ac80-419d6855ffc4" name="START15"
  x="795" y="570" z="0"
  width="50" height="85"
  startAngle="0" endAngle="6.283185308"
  startDate="0" endDate=""
  place="05bde813-aeca-472c-8395-ce4bbbb8d47f">
```

```
  <entity budget="40"
    agentType="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
    type="fr.utbm.set.jasim.environment.semantics.PedestrianType">
    <frustums>
    <frustum type="pedestrian" eyePosition="1.8" farDistance="100"
      nearDistance="10" fieldOfViewHAngle="0.785398164"
      fieldOfViewVAngle="0.785398164"/>
    </frustums>
    <body type="" height="" radius="" />
    <generationLaw class="fr.utbm.set.jasim.spawn.ConstantSpawningLaw">
      <lawParam name="value" value="1000000000000" />
    </generationLaw>
    <goals>
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="567.5" y="660" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    <goal activate="true" place="c46a25d0-8663-11e0-9d78-0800200c9a66"
      x="540" y="1180" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    </goals>
  </entity>
</spawner>
<spawner type="area" id="116b5fb4-432a-4948-ac80-419d6855ffc5" name="START16"
  x="500" y="400" z="0"
  width="800" height="150"
  startAngle="0" endAngle="6.283185308"
  startDate="0" endDate=""
  place="05bde813-aeca-472c-8395-ce4bbbb8d47f">
  <entity budget="40"
    agentType="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
    type="fr.utbm.set.jasim.environment.semantics.PedestrianType">
    <frustums>
    <frustum type="pedestrian" eyePosition="1.8" farDistance="100"
      nearDistance="10" fieldOfViewHAngle="0.785398164"
      fieldOfViewVAngle="0.785398164"/>
    </frustums>
    <body type="" height="" radius="" />
    <generationLaw class="fr.utbm.set.jasim.spawn.ConstantSpawningLaw">
      <lawParam name="value" value="1000000000000" />
    </generationLaw>
    <goals>
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="1232.5" y="660" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    <goal activate="true" place="ef98aec0-8663-11e0-9d78-0800200c9a66"
      x="1320" y="1180" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    </goals>
  </entity>
</spawner>
<spawner type="area" id="116b5fb4-432a-4948-ac80-419d6855ffc6" name="START17"
  x="500" y="400" z="0"
  width="800" height="150"
  startAngle="0" endAngle="6.283185308"
  startDate="0" endDate=""
  place="05bde813-aeca-472c-8395-ce4bbbb8d47f">
  <entity budget="40"
    agentType="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
    type="fr.utbm.set.jasim.environment.semantics.PedestrianType">
    <frustums>
```

```
    <frustum type="pedestrian" eyePosition="1.8" farDistance="100"
      nearDistance="10" fieldOfViewHAngle="0.785398164"
      fieldOfViewVAngle="0.785398164"/>
    </frustums>
    <body type="" height="" radius="" />
    <generationLaw class="fr.utbm.set.jasim.spawn.ConstantSpawningLaw">
      <lawParam name="value" value="1000000000000" />
    </generationLaw>
    <goals>
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="1232.5" y="660" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    <goal activate="true" place="ef98aec0-8663-11e0-9d78-0800200c9a66"
      x="1320" y="1180" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    </goals>
  </entity>
</spawner>
<spawner type="area" id="116b5fb4-432a-4948-ac80-419d6855ffc7" name="START18"
  x="500" y="400" z="0"
  width="800" height="150"
  startAngle="0" endAngle="6.283185308"
  startDate="0" endDate=""
  place="05bde813-aeca-472c-8395-ce4bbbb8d47f">
  <entity budget="40"
    agentType="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
    type="fr.utbm.set.jasim.environment.semantics.PedestrianType">
    <frustums>
    <frustum type="pedestrian" eyePosition="1.8" farDistance="100"
      nearDistance="10" fieldOfViewHAngle="0.785398164"
      fieldOfViewVAngle="0.785398164"/>
    </frustums>
    <body type="" height="" radius="" />
    <generationLaw class="fr.utbm.set.jasim.spawn.ConstantSpawningLaw">
      <lawParam name="value" value="1000000000000" />
    </generationLaw>
    <goals>
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="567.5" y="660" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    <goal activate="true" place="c46a25d0-8663-11e0-9d78-0800200c9a66"
      x="540" y="1180" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    </goals>
  </entity>
</spawner>
<spawner type="area" id="116b5fb4-432a-4948-ac80-419d6855ffc8" name="START19"
  x="500" y="400" z="0"
  width="800" height="150"
  startAngle="0" endAngle="6.283185308"
  startDate="0" endDate=""
  place="05bde813-aeca-472c-8395-ce4bbbb8d47f">
  <entity budget="40"
    agentType="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
    type="fr.utbm.set.jasim.environment.semantics.PedestrianType">
    <frustums>
    <frustum type="pedestrian" eyePosition="1.8" farDistance="100"
      nearDistance="10" fieldOfViewHAngle="0.785398164"
      fieldOfViewVAngle="0.785398164"/>
    </frustums>
```

```
    <body type="" height="" radius="" />
    <generationLaw class="fr.utbm.set.jasim.spawn.ConstantSpawningLaw">
      <lawParam name="value" value="1000000000000" />
    </generationLaw>
    <goals>
    <goal activate="true" place="05bde813-aeca-472c-8395-ce4bbbb8d47f"
      x="1232.5" y="660" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    <goal activate="true" place="ef98aec0-8663-11e0-9d78-0800200c9a66"
      x="1320" y="1180" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    </goals>
  </entity>
</spawner>
<spawner type="area" id="116b5fb4-432a-4948-ac80-419d6855ffd0" name="START20"
  x="1128" y="690" z="0"
  width="170" height="45"
  startAngle="0" endAngle="6.283185308"
  startDate="0" endDate=""
  place="05bde813-aeca-472c-8395-ce4bbbb8d47f">
  <entity budget="60"
    agentType="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
    type="fr.utbm.set.jasim.environment.semantics.PedestrianType">
    <frustums>
    <frustum type="pedestrian" eyePosition="1.8" farDistance="100"
      nearDistance="10" fieldOfViewHAngle="0.785398164"
      fieldOfViewVAngle="0.785398164"/>
    </frustums>
    <body type="" height="" radius="" />
    <generationLaw class="fr.utbm.set.jasim.spawn.ConstantSpawningLaw">
      <lawParam name="value" value="1000000000000" />
    </generationLaw>
    <goals>
    <goal activate="true" place="ef98aec0-8663-11e0-9d78-0800200c9a66"
      x="1320" y="1180" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    </goals>
  </entity>
</spawner>
<spawner type="area" id="116b5fb4-432a-4948-ac80-419d6855ffd7" name="START21"
  x="474" y="690" z="0"
  width="170" height="45"
  startAngle="0" endAngle="6.283185308"
  startDate="0" endDate=""
  place="05bde813-aeca-472c-8395-ce4bbbb8d47f">
  <entity budget="60"
    agentType="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
    type="fr.utbm.set.jasim.environment.semantics.PedestrianType">
    <frustums>
    <frustum type="pedestrian" eyePosition="1.8" farDistance="100"
      nearDistance="10" fieldOfViewHAngle="0.785398164"
      fieldOfViewVAngle="0.785398164"/>
    </frustums>
    <body type="" height="" radius="" />
    <generationLaw class="fr.utbm.set.jasim.spawn.ConstantSpawningLaw">
      <lawParam name="value" value="1000000000000" />
    </generationLaw>
    <goals>
    <goal activate="true" place="c46a25d0-8663-11e0-9d78-0800200c9a66"
```

```
      x="540" y="1180" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    </goals>
  </entity>
</spawner>


<!-- Spawner in board hall one for initialisation -->
<spawner type="area" id="116b5fb4-432a-4948-ac80-419d6855ffd1" name="START22"
  x="770" y="1070" z="0"
  width="1000" height="85"
  startAngle="0" endAngle="6.283185308"
  startDate="0" endDate=""
  place="ef98aec0-8663-11e0-9d78-0800200c9a66">
  <entity budget="100"
    agentType="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
    type="fr.utbm.set.jasim.environment.semantics.PedestrianType">
    <frustums>
    <frustum type="pedestrian" eyePosition="1.8" farDistance="100"
      nearDistance="10" fieldOfViewHAngle="0.785398164"
      fieldOfViewVAngle="0.785398164"/>
    </frustums>
    <body type="" height="" radius="" />
    <generationLaw class="fr.utbm.set.jasim.spawn.ConstantSpawningLaw">
      <lawParam name="value" value="1000000000000" />
    </generationLaw>
    <goals>
    <goal activate="true" place="ef98aec0-8663-11e0-9d78-0800200c9a66"
      x="1320" y="1180" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    </goals>
  </entity>
</spawner>
<spawner type="area" id="116b5fb4-432a-4948-ac80-419d6855ffd2" name="START23"
  x="770" y="1070" z="0"
  width="1000" height="85"
  startAngle="0" endAngle="6.283185308"
  startDate="0" endDate=""
  place="ef98aec0-8663-11e0-9d78-0800200c9a66">
  <entity budget="100"
    agentType="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
    type="fr.utbm.set.jasim.environment.semantics.PedestrianType">
    <frustums>
    <frustum type="pedestrian" eyePosition="1.8" farDistance="100"
      nearDistance="10" fieldOfViewHAngle="0.785398164"
      fieldOfViewVAngle="0.785398164"/>
    </frustums>
    <body type="" height="" radius="" />
    <generationLaw class="fr.utbm.set.jasim.spawn.ConstantSpawningLaw">
      <lawParam name="value" value="1000000000000" />
    </generationLaw>
    <goals>
    <goal activate="true" place="ef98aec0-8663-11e0-9d78-0800200c9a66"
      x="1320" y="1180" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    </goals>
  </entity>
</spawner>
<spawner type="area" id="116b5fb4-432a-4948-ac80-419d6855ffd3" name="START24"
```

```
  x="770" y="1070" z="0"
  width="1000" height="85"
  startAngle="0" endAngle="6.283185308"
  startDate="0" endDate=""
  place="ef98aec0-8663-11e0-9d78-0800200c9a66">
  <entity budget="100"
    agentType="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
    type="fr.utbm.set.jasim.environment.semantics.PedestrianType">
    <frustums>
    <frustum type="pedestrian" eyePosition="1.8" farDistance="100"
      nearDistance="10" fieldOfViewHAngle="0.785398164"
      fieldOfViewVAngle="0.785398164"/>
    </frustums>
    <body type="" height="" radius="" />
    <generationLaw class="fr.utbm.set.jasim.spawn.ConstantSpawningLaw">
      <lawParam name="value" value="1000000000000" />
    </generationLaw>
    <goals>
    <goal activate="true" place="ef98aec0-8663-11e0-9d78-0800200c9a66"
      x="1320" y="1180" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    </goals>
  </entity>
</spawner>
<spawner type="area" id="116b5fb4-432a-4948-ac80-419d6855ffd4" name="START25"
  x="770" y="1070" z="0"
  width="1000" height="85"
  startAngle="0" endAngle="6.283185308"
  startDate="0" endDate=""
  place="ef98aec0-8663-11e0-9d78-0800200c9a66">
  <entity budget="100"
    agentType="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
    type="fr.utbm.set.jasim.environment.semantics.PedestrianType">
    <frustums>
    <frustum type="pedestrian" eyePosition="1.8" farDistance="100"
      nearDistance="10" fieldOfViewHAngle="0.785398164"
      fieldOfViewVAngle="0.785398164"/>
    </frustums>
    <body type="" height="" radius="" />
    <generationLaw class="fr.utbm.set.jasim.spawn.ConstantSpawningLaw">
      <lawParam name="value" value="1000000000000" />
    </generationLaw>
    <goals>
    <goal activate="true" place="ef98aec0-8663-11e0-9d78-0800200c9a66"
      x="1320" y="1180" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    </goals>
  </entity>
</spawner>
<spawner type="area" id="116b5fb4-432a-4948-ac80-419d6855ffd5" name="START26"
  x="770" y="1070" z="0"
  width="1000" height="85"
  startAngle="0" endAngle="6.283185308"
  startDate="0" endDate=""
  place="ef98aec0-8663-11e0-9d78-0800200c9a66">
  <entity budget="100"
    agentType="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
    type="fr.utbm.set.jasim.environment.semantics.PedestrianType">
```

```
    <frustums>
    <frustum type="pedestrian" eyePosition="1.8" farDistance="100"
      nearDistance="10" fieldOfViewHAngle="0.785398164"
      fieldOfViewVAngle="0.785398164"/>
    </frustums>
    <body type="" height="" radius="" />
    <generationLaw class="fr.utbm.set.jasim.spawn.ConstantSpawningLaw">
      <lawParam name="value" value="1000000000000" />
    </generationLaw>
    <goals>
    <goal activate="true" place="ef98aec0-8663-11e0-9d78-0800200c9a66"
      x="1320" y="1180" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    </goals>
  </entity>
</spawner>
<spawner type="area" id="116b5fb4-432a-4948-ac80-419d6855ffd6" name="START27"
  x="770" y="1070" z="0"
  width="1000" height="85"
  startAngle="0" endAngle="6.283185308"
  startDate="0" endDate=""
  place="ef98aec0-8663-11e0-9d78-0800200c9a66">
  <entity budget="100"
    agentType="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
    type="fr.utbm.set.jasim.environment.semantics.PedestrianType">
    <frustums>
    <frustum type="pedestrian" eyePosition="1.8" farDistance="100"
      nearDistance="10" fieldOfViewHAngle="0.785398164"
      fieldOfViewVAngle="0.785398164"/>
    </frustums>
    <body type="" height="" radius="" />
    <generationLaw class="fr.utbm.set.jasim.spawn.ConstantSpawningLaw">
      <lawParam name="value" value="1000000000000" />
    </generationLaw>
    <goals>
    <goal activate="true" place="ef98aec0-8663-11e0-9d78-0800200c9a66"
      x="1320" y="1180" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    </goals>
  </entity>
</spawner>



<!-- Spawners in board hall two for initialisation -->
<spawner type="area" id="116b5fb4-432a-4948-ac80-419d6855ffd8" name="START28"
  x="50" y="1070" z="0"
  width="500" height="85"
  startAngle="0" endAngle="6.283185308"
  startDate="0" endDate=""
  place="c46a25d0-8663-11e0-9d78-0800200c9a66">
  <entity budget="100"
    agentType="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
    type="fr.utbm.set.jasim.environment.semantics.PedestrianType">
    <frustums>
    <frustum type="pedestrian" eyePosition="1.8" farDistance="100"
      nearDistance="10" fieldOfViewHAngle="0.785398164"
      fieldOfViewVAngle="0.785398164"/>
```

```
    </frustums>
    <body type="" height="" radius="" />
    <generationLaw class="fr.utbm.set.jasim.spawn.ConstantSpawningLaw">
      <lawParam name="value" value="1000000000000" />
    </generationLaw>
    <goals>
    <goal activate="true" place="c46a25d0-8663-11e0-9d78-0800200c9a66"
      x="540" y="1180" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    </goals>
  </entity>
</spawner>
<spawner type="area" id="116b5fb4-432a-4948-ac80-419d6855ffd9" name="START29"
  x="50" y="1070" z="0"
  width="500" height="85"
  startAngle="0" endAngle="6.283185308"
  startDate="0" endDate=""
  place="c46a25d0-8663-11e0-9d78-0800200c9a66">
  <entity budget="100"
    agentType="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
    type="fr.utbm.set.jasim.environment.semantics.PedestrianType">
    <frustums>
    <frustum type="pedestrian" eyePosition="1.8" farDistance="100"
      nearDistance="10" fieldOfViewHAngle="0.785398164"
      fieldOfViewVAngle="0.785398164"/>
    </frustums>
    <body type="" height="" radius="" />
    <generationLaw class="fr.utbm.set.jasim.spawn.ConstantSpawningLaw">
      <lawParam name="value" value="1000000000000" />
    </generationLaw>
    <goals>
    <goal activate="true" place="c46a25d0-8663-11e0-9d78-0800200c9a66"
      x="540" y="1180" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    </goals>
  </entity>
</spawner>
<spawner type="area" id="116b5fb4-432a-4948-ac80-419d6855ffe0" name="START30"
  x="50" y="1070" z="0"
  width="500" height="85"
  startAngle="0" endAngle="6.283185308"
  startDate="0" endDate=""
  place="c46a25d0-8663-11e0-9d78-0800200c9a66">
  <entity budget="100"
    agentType="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
    type="fr.utbm.set.jasim.environment.semantics.PedestrianType">
    <frustums>
    <frustum type="pedestrian" eyePosition="1.8" farDistance="100"
      nearDistance="10" fieldOfViewHAngle="0.785398164"
      fieldOfViewVAngle="0.785398164"/>
    </frustums>
    <body type="" height="" radius="" />
    <generationLaw class="fr.utbm.set.jasim.spawn.ConstantSpawningLaw">
      <lawParam name="value" value="1000000000000" />
    </generationLaw>
    <goals>
    <goal activate="true" place="c46a25d0-8663-11e0-9d78-0800200c9a66"
      x="540" y="1180" z="0" tangentX="0" tangentY="1" tangentZ="0" />
```

```
    </goals>
  </entity>
</spawner>
<spawner type="area" id="116b5fb4-432a-4948-ac80-419d6855ffe1" name="START31"
  x="50" y="1070" z="0"
  width="500" height="85"
  startAngle="0" endAngle="6.283185308"
  startDate="0" endDate=""
  place="c46a25d0-8663-11e0-9d78-0800200c9a66">
  <entity budget="100"
    agentType="fr.utbm.set.demo.simulation.pedestrians.PedestrianHolon"
    type="fr.utbm.set.jasim.environment.semantics.PedestrianType">
    <frustums>
    <frustum type="pedestrian" eyePosition="1.8" farDistance="100"
      nearDistance="10" fieldOfViewHAngle="0.785398164"
      fieldOfViewVAngle="0.785398164"/>
    </frustums>
    <body type="" height="" radius="" />
    <generationLaw class="fr.utbm.set.jasim.spawn.ConstantSpawningLaw">
      <lawParam name="value" value="1000000000000" />
    </generationLaw>
    <goals>
    <goal activate="true" place="c46a25d0-8663-11e0-9d78-0800200c9a66"
      x="540" y="1180" z="0" tangentX="0" tangentY="1" tangentZ="0" />
    </goals>
  </entity>
</spawner>
</spawners>

<renderers>
<renderer type="user" class="fr.utbm.set.MyRenderer"/>
</renderers>

</simulation>
```

E

SYNCHRONISATION DES MODÈLES DE SIMULATION

L'un des problèmes majeur de la simulation distribuée est la synchronisation des différents modèles [Filloque, 1992]. En effet, il faut s'assurer que les entités transitant d'un modèle vers un autre arrivent ni trop tôt, ni trop tard. Cela se traduit par le fait que le modèle recevant l'entité a une date de simulation inférieure ou égale à celle à laquelle l'unité doit arriver dans ce système. De manière plus formelle, K. Mani CHANDY, Jayadev MISRA et Randall BRYANT ont répondu aux problèmes de synchronisation par la définition de la notion de causalité [Bryant, 1979, Chandy et Misra, 1979, 1988] :

Définition E.1 Causalité

La causalité est le principe de non influence du futur sur le passé. Si cette contrainte est respectée localement et que les interactions se font uniquement par messages estampillés, alors la simulation traite les événements dans un ordre consistant avec les ordres partiels imposés par la contrainte de causalité du système réel.

Notons qu'un message est un événement transitant entre deux modèles distribués. De plus, une seconde contrainte doit être respectée pour que la synchronisation des modèles de simulation puisse être effectuée. Cette nouvelle contrainte, dite de vivacité, est définie comme suit :

Définition E.2 Vivacité

La vivacité est la contrainte imposant au temps de toujours progresser. Son respect permet à une simulation de ne pas se trouver en situation de blocage.

Dans la suite de cette section, nous présentons les trois approches majeures utilisées pour que la causalité soit respectée. Dans ces descriptions, nous utilisons la notation suivante :

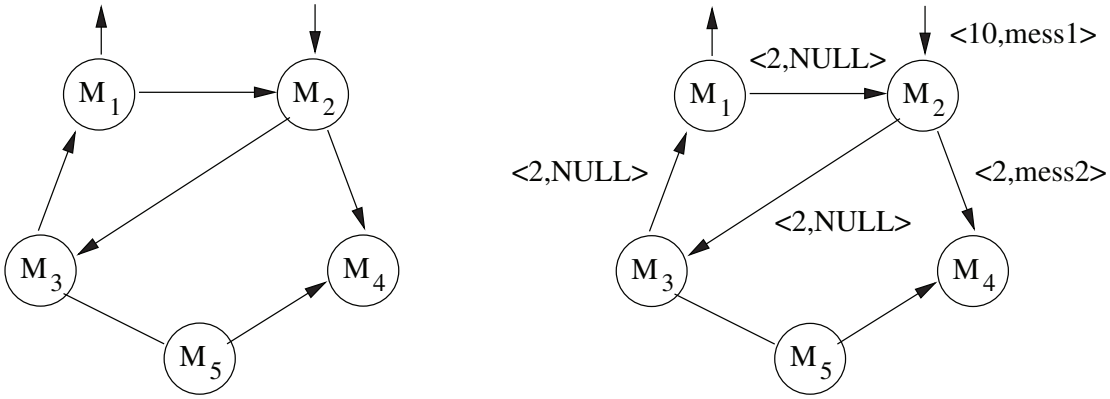
- les événements sont définis par le doublet $\langle t_r, type \rangle$ où t_r représente la date d'occurrence de l'événement et $type$ sa description ;
- la notion d'horloge locale à un modèle, notée TVL, correspond au minimum des estampilles des événements en attente d'un traitement : $\text{TVL} = \min(\{\forall e, tr_e\})$ où tr_e est l'estampille de l'événement entrant e ;
- Pour respecter la contrainte de causalité, l'horloge commune à tous les modèles (notée TVG) est définie comme le minimum des horloges locales : $\text{TVG} = \min(\{\forall m, \text{TVL}_m\})$ où $textsctvl_m$ est l'horloge locale du modèle m.

Les trois approches de synchronisation reconnues par la communauté scientifique sont les approches pessimiste, optimiste et hybride.

E.1/ APPROCHE PESSIMISTE

Dans cette approche, pour vérifier la contrainte de causalité, un modèle m ne doit traiter un événement e que s'il est certain qu'il ne recevra pas des autres modèles un message porteur d'une date inférieure. Ce principe est qualifié de pessimiste par K. Mani CHANDY et Jayadev MISRA [Chandy et Misra, 1979].

Ce principe de sûreté peut provoquer des situations de blocage. Ces dernières se produisent lorsqu'un modèle ne reçoit pas de messages sur ses entrées. Par exemple, la figure E.1 illustre un système composé de cinq modèles. Ce système possédant un cycle entre M_1, M_2 et M_3, le modèle M_4 peut rester bloqué dans l'attente d'un message en provenance de M_5 (la contrainte de vivacité n'est alors plus respectée). Ainsi, la différence entre les différents algorithmes pessimistes réside essentiellement dans l'approche de satisfaction de la contrainte de vivacité.



(a) Topologie de modèles distribués (b) Situation de blocage

FIGURE E.1 – Synchronisation - Exemples

K. Mani CHANDY et Jayadev MISRA proposent un algorithme de prévention des blocages. Pour cela, il emploie des messages supplémentaires appelés NULL. Lors de la simulation, un modèle doit envoyer, dès réception d'un événement, un message sur tous ses canaux de sortie. S'il ne peut émettre un message utile, il doit envoyer un message NULL contenant sa TVL. Ainsi, un autre modèle pourra décider de traiter un événement grâce aux informations que lui procurent sa propre TVL et la borne inférieure des temps de ses voisins. [Chandy et Misra, 1979] montre que l'utilisation des messages NULL avec une estampille correspondant à la TVL ne permet pas de résoudre complètement le problème de satisfaction de la contrainte de vivacité. La figure E.1 illustre un exemple de blocage provoqué par l'utilisation des messages NULL. La réception par M_3 d'un message $\langle 2, \text{NULL} \rangle$ provenant de M_2 ne lui permet pas d'avancer. Par conséquent, il émet le message $\langle 2, \text{NULL} \rangle$ vers M_1. Ce dernier ne pouvant pas non plus avancer, il émet le même message vers M_2. De cette situation émerge un blocage entre M_1, M_2 et M_3. Ce problème est éliminé en introduisant la notion de prédiction (notée σ). Cette valeur locale à chaque modèle est strictement positive et représente la quantité de temps futur pour lequel le modèle peut prévoir son comportement, c'est-à-dire les messages qu'il enverra. Cette nouvelle contrainte de *prédictabilité* impose qu'à chaque envoi d'un message NULL, l'estampille soit $\min(\{\forall e, tr_e\}) + \sigma$ pour tout événement e défini par $\langle tr_e, type_e \rangle$.

[Chandy et Misra, 1981] proposent une autre approche pessimiste : la guérison de blocage. Contrairement à la précédente, elle n'utilise pas les messages NULL. Chaque modèle consomme l'événement en entrée qui possède la plus petite estampille. Si la liste est vide, le site se bloque. Cette technique permet de ne pas violer la contrainte de causalité. Par contre, celle de vivacité peut être transgressée. Un mécanisme associé au simulateur permet de détecter les blocages et de relancer les sites correspondants. Pour cela, les étapes suivantes sont appliquées :

1. simulation jusqu'à une situation de blocage,

2. détection du blocage,
3. destruction du blocage en relançant le ou les modèles qui ont la plus petite estampille des événements en entrée.

Cette approche utilise un algorithme dérivé de celui proposé par Edsger DIJKSTRA et Carel SCHOLTEN [Dijkstra et Scholten, 1980]. Malheureusement, cet algorithme admet des situations de blocage sur les sorties notamment à cause de l'hypothèse de files bornées. [Ingels et Raynal, 1990] proposent une solution simplifiée grâce à l'hypothèse des files infinies.

E.2/ APPROCHE OPTIMISTE

Un protocole optimiste n'impose aucune contrainte sur le comportement des modèles [Fujimoto, 1990a,b]. Il laisse les erreurs de causalité se produire, les détecte et les répare. Un modèle peut avancer dans le temps aussi vite qu'il le peut, consommant tous les événements présents en entrée, et en faisant le pari qu'il ne se produira aucune erreur. La vivacité est garantie car les modèles n'attendent jamais, leur TVL est incrémentée jusqu'à la fin de la simulation. Une violation de la contrainte de causalité est détectée quand l'estampille d'un événement arrivant est inférieure à la TVL. Un mécanisme est alors mis en œuvre pour annuler toutes les actions réalisées entre la date de l'événement et la TVL courante. Cette approche nécessite la sauvegarde de l'ensemble des états de la simulation. Par rapport aux protocoles pessimistes, on fait l'hypothèse que le temps passé à défaire les traitements erronés sera inférieur à celui nécessaire pour s'assurer que l'on pouvait avancer. Un espace de sauvegarde suffisant est la seconde hypothèse.

David JEFFERSON et Henry SOWIZRAL sont les premiers a avoir proposé un algorithme optimiste de synchronisation sous le nom de *« Time Warp »* [Jefferson, 1985, Jefferson et Sowizral, 1985]. Le principe est de diffuser des *anti-messages* pour annuler les effets distants des événements erronés. Dans cette approche les événements sont définis par $\langle te, te, S_e, S_r, s\rangle$ où :
– te est la TVL de l'émetteur, et est utile pour le mécanisme de retour arrière ;
– tr est équivalent à l'estampille de l'événement (date d'arrivée de l'événement) ;
– S_e est le modèle émetteur ;
– S_r est le modèle récepteur ;
– s est le signe du message : $\oplus$ indique un message *« normal »* et $\ominus$ un anti-message.
Lors de l'arrivée d'un message, le modèle compare la date d'estampille tr à sa propre TVL :

if $tr <$ TVL **then**
le message est simplement placé dans la file d'entrée et sera traité à son tour
else
les traitements effectués entre tr et TVL sont erronés ; il faut faire un retour en arrière qui conduit aux actions suivantes :

1. annuler les traitements locaux en restaurant un des vecteurs d'état sauvegardés avec une date inférieure à tr ;
2. annuler les émissions de messages réalisées après tr. Trois cas sont envisageables :
   (a) le message à annuler n'est pas encore émis : il suffit de le supprimer de la file de sortie ;
   (b) le message émis n'a pas encore été traité par le récepteur : ce dernier le retire de sa file d'entrée ;
   (c) le message a été consommé : le récepteur doit réaliser le traitement de retour en arrière à partir de la première étape ;
3. annuler les demandes d'actions définitives générées après tr.

end if

Cette approche fait l'hypothèse de l'utilisation d'un espace mémoire infini. Passer à une représentation bornée, plus réaliste pour les systèmes réels, nécessite de pouvoir limiter les besoins pour

récupérer l'espace inutile. [Samadi, 1985] prouve l'existence d'une borne inférieure en dessous de laquelle le système ne peut jamais revenir. Elle permet d'oublier les états correspondant aux dates inférieures.

De nombreuses variantes ont été étudiées pour limiter l'effet négatif des retours arrière.

L'*annulation paresseuse* permet de retarder l'émission des anti-messages tant que l'on n'est pas sûr que les messages émis soient réellement en faute. Cette approche a pour origine le fait que la ré-exécution peut conduire en partie aux mêmes conséquences que l'exécution en faute [Gafni, 1988, Reiher, 1990], et *« qu'un bon événement peut être généré par une erreur »* [Berry, 1986]. Le principal inconvénient de cette approche est le retard apporté à la correction des erreurs réelles qui permet aux effets dévastateurs de résider plus longtemps dans la simulation.

[Sokol et al., 1988] propose une autre solution, appelée *« Moving Time Window »* , qui consiste à limiter l'optimisme de l'exécution pour empêcher les erreurs de trop se propager. Pour cela, une *fenêtre* fixant un horizon au delà duquel la TVL ne peut aller est fixée. Cette démarche est critiquée car il est difficile de déterminer la bonne taille de la fenêtre, mais aussi parce qu'un gel de la simulation peut se produire malgré une évolution correcte.

[Madisetti et al., 1988] propose une optimisation appelée *« Wolf Call »* qui consiste à envoyer un message spécial gelant tous les modèles qui ont été perturbés par l'erreur. Cette technique nécessite toutefois une transmission rapide mais aussi la maîtrise des délais de transmission en temps réel pour déterminer la sphère d'influence des messages erronés.

E.3/ APPROCHE HYBRIDE

Paul REYNOLDS soutient que la dichotomie dans le domaine de la simulation distribuée entre l'approche pessimiste et l'approche hybride est une erreur [Reynolds, 1988]. Il pense qu'il existe un grand nombre de choix dont les deux extrêmes sont les approches pessimiste et optimiste.

Une approche couramment utilisée est l'ajout du comportement opposé à celui choisi comme base.

Par exemple, sur une base pessimiste, il est possible de plaquer un certain optimisme en calculant les événements sûrs puis, par précaution, les événements moins sûrs. Dans ce groupe, on peut trouver :

- un algorithme proposé par Boris LUBACHEVSKY, Adam SHWARTZ et Alan WEISS qui permet de traiter les événements non sûrs en fonction de connaissances données par l'utilisateur [Lubachevsky et al., 1989] ;
- une méthode de calculs spéculatifs qui sont réalisés sans envoi de message pouvant générer des erreurs. Les émissions sont alors exécutées une fois la validation acquise [Mehl, 1991] ;
- une proposition de Paul REYNOLDS et Philip DICKENS appellée SRAD/LR et proche de la méthode précédente. Elle permet de limiter l'agressivité en calculant dynamiquement la limite supérieure d'optimisme [Reynolds et Dickens, 1990] ;
- une approche par calculs spéculatifs avec émission de messages, mais avec une fenêtre limitée statistiquement [Sokol et al., 1988].

D'autres propositions consistent à dégrader le modèle optimiste en calculant dynamiquement des fenêtres en fonction du comportement de chaque modèle.

Une approche originale, *« COMPOSITE ELSA »* , consiste à rechercher a priori les parties de modèles pour lesquelles l'approche pessimiste est préférable et d'autres parties pour lesquelles l'optimisme serait plus adapté [Arvind et Smart, 1992, Arvind et al., 1991].

Résumé :

Cette thèse propose un modèle organisationnel et holonique de l'environnement pour la simulation des déplacements de piétons dans des bâtiments. Une foule de piétons peut être considérée comme un système composé d'un grand nombre d'entités en interaction, dont la dynamique globale ne peut se réduire à la somme des comportements de ses composants. La simulation multiniveau fondée sur les modèles multiagents holoniques constitue une approche permettant d'analyser la dynamique de tels systèmes. Elle autorise leur analyse en considérant plusieurs niveaux d'observation (microscopique, mésoscopique et macroscopique) et prend en compte les ressources de calcul disponibles. Dans ces systèmes, l'environnement est considéré comme l'une des parties essentielles. La dynamique des piétons composant la foule est alors clairement distinguée de celle de l'environnement dans lequel ils se déplacent. Un modèle organisationnel décrivant la structure et la dynamique de l'environnement est proposé. L'environnement est structurellement décomposé en zones, sous-zones, *etc*. Les organisations et les rôles sont projetés dans une société d'agents ayant en charge de simuler la dynamique de l'environnement et les différentes missions qui lui sont classiquement assignées dans les systèmes multiagents. Ce modèle précise également les règles de passage entre deux niveaux d'observation. Ainsi, chaque agent appartenant au modèle de l'environnement tente d'utiliser une approximation des comportements de ses sous-zones afin de limiter la consommation de ressources durant la simulation. La qualité de l'approximation entre ces deux niveaux d'observation est évaluée avec des indicateurs énergétiques. Ils permettent de déterminer si l'agent approxime correctement les comportements des agents associés aux sous-zones. En sus du modèle organisationnel et holonique proposé, nous présentons un modèle concret de la simulation de voyageurs dans un terminal d'aéroport. Ce modèle concret est implanté sur les plateformes JASIM et JANUS.

Mots-clés : Modèle d'environnement, Simulation multiniveau, Systèmes multiagents, Approche organisationnelle, Simulation de foules

Abstract:

This work presents a holonic organizational model of the environment for the simulation of pedestrians in buildings. A crowd of pedestrians is considered as a system composed of a large number of interacting entities. The global dynamics of this system cannot be reduced to the sum of the behaviors of its components. Multilevel simulation based on holonic multiagent models is one approach to analyze the dynamics of such systems. It allows their analysis by considering several levels of observation (microscopic, mesoscopic and macroscopic) and the available computing resources. In these systems, the environment is considered as an essential part. The behavior of the crowd is clearly distinguished from the behavior of the environment in which the pedestrians move. An organizational model is proposed to describe the structure and the dynamics of the indoor environment. This environment is structurally divided into areas, sub-areas, *etc*. Organizations and roles are mapped into a society of agents in charge of simulating the dynamics of the environment and their various missions in multiagent systems. This model also specifies the rules for changing the level of observation dynamically. Thus, each agent belonging to the model of the environment tries to use an approximation of behaviors of its sub-zones, and at the same time to minimize the resource consumption. The quality of the approximation between these two levels is evaluated with energy-based indicators. They help to determine if the agent approximates the behaviors of its sub-agents correctly. In addition to the organizational and holonic model proposed in this work, we present a concrete model of the simulation of passengers in an airport terminal. This concrete model is implemented on the platforms JASIM and JANUS.

Keywords: Environment model, Multilevel simulation, Multiagents systems, Organizational approach, Crowd simulations

SPIM

■ École doctorale SPIM - Université de Technologie Belfort-Montbéliard
F - 90010 Belfort Cedex ■ tél. +33 (0)3 84 58 31 39
■ ed-spim@univ-fcomte.fr ■ www.ed-spim.univ-fcomte.fr

utbm
université de technologie Belfort-Montbéliard