Vers une représentation des comportements centrée interactions

Towards an interaction-based design of behaviors

Ph. Mathieu* S. Picault*

*LIFL (CNRS UMR 8022)

Laboratoire d'Informatique Fondamentale de Lille
Université des Sciences et Technologies de Lille
Cité Scientifique – 59655 Villeneuve d'Ascq Cedex
{mathieu, picault}@lifl.fr

Résumé

Dans cet article nous défendons les avantages d'une représentation des comportements d'agents basée sur la notion d'interaction. Alors que d'ordinaire le comportement d'un agent est inclus dans sa structure, nous dissocions ici la notion d'agent et celle d'interaction, chacune pouvant faire l'objet d'une ontologie propre. Cette approche a notamment pour avantage de faciliter la réutilisabilité des interactions qui sont bien souvent génériques dans des contextes où les agents ne le sont pas. Nous montrons qu'elle est particulièrement bien adaptée à la simulation large échelle tant du point de vue du nombre d'individus que de la diversité des comportements, et proposons la méthodologie IODA, à la frontière du génie logiciel et de la représentation des connaissances pour les systèmes de simulation à agents.

Mots Clef

Simulation, Interaction, Comportement

Abstract

In this paper we defend the advantages of a representation of agent behaviors, based on the concept of interaction. Although the behavior of an agent is usually included in its structure, we dissociate here the concept of agent from that of interaction, each one being associated with a specific ontology. This approach is especially valuable for increasing the reutilisability of the interactions, which are very often generic even in contexts where the agents are not. We show here that this approach is particularly well adapted to broad simulation scales, regarding as well the number of individuals as the diversity of the behaviors, and propose the IODA methodology, at the intersection between software engineering and knowledge representation in agent simulation systems.

Keywords

Simulation, Interaction, Behavior

1 Introduction

Depuis quelques années, la simulation à base d'agents a pris une place prépondérante dans les outils de reproduction du vivant, que ce soit pour en comprendre les mécanismes ou pour en reproduire les caractéristiques à des fins ludiques (jeux vidéo, animation cinématographique, etc.). La réalisation d'une simulation pose d'emblée des problèmes qui relèvent à la fois de l'implémentation (i.e. du génie logiciel) et de la modélisation (i.e. de la représentation de connaissances).

Si l'on examine d'abord les outils permettant de développer une simulation, on dispose d'un grand nombre de plateformes multi-agents ouvertes, c'est-à-dire qui laissent au concepteur une totale liberté de codage des agents, de leurs comportements, de l'environnement. On peut les caractériser par un plus ou moins grand degré de sophistication logicielle. Il existe en particulier de nombreux frameworks multi-agents pour lesquels tous les raffinements du génie logiciel sont envisageables (réutilisation, généricité, *design patterns*, composants). C'est le cas notamment de Swarm [1] ou de Madkit [2]. Ces plateformes, d'ailleurs, ne sont pas spécifiquement dédiées à la réalisation de *simulations*, mais peuvent également être employées par exemple pour de la résolution distribuée de problèmes. Leur intérêt réside principalement dans la réutilisabilité du code, développé dans un contexte logiciel générique.

D'autres outils, comme Netlogo [3], sont au contraire conçus pour pouvoir être utilisées par des non-informaticiens, et reposent donc sur une forme très simple de programmation. En l'occurrence, Netlogo permet de piloter en parallèle des « tortues » qui exécutent des procédures, sans préjuger du domaine d'application des simulations réalisées. L'ouverture de ces plateformes et leur généricité sont malheureusement obtenues au

détriment d'un cadre précis de modélisation qui guiderait la conception des comportements. Ainsi par exemple NetLogo ne propose pas d'abstraction ; la notion de comportement n'est même pas réifiée.

Du point de vue de l'analyse-conception, divers formalismes permettant d'exprimer des comportements (subsomption, réseaux neuronaux, réseaux de Petri, bases de règles...), constituent un cadre suffisamment précis pour guider fortement la modélisation, au détriment dans ce cas de la réutilisabilité des comportements, comme on peut le constater dans la majorité des simulations dédiées à l'étude d'un phénomène particulier. Les comportements d'agents développés pour une simulation, s'ils sont exprimés dans le cadre d'une architecture donnée, ne peuvent être réutilisés qu'avec la même architecture ; or le choix de cette dernière est souvent fortement lié aux spécifités du problème traité. L'une des rares architectures notables de cette catégorie, permettant une généricité des comportements quel que soit le domaine d'application, est le modèle BDI (*Belief-Desire-Intention*) ou ses variantes, dans la mesure où il sépare les connaissances de leur traitement ; mais il est souvent inadéquat pour la réalisation de simulations.

Chacun conviendra que l'idéal serait d'opérer une synthèse de ces approches, permettant aussi bien d'un point de logiciel que d'un point de vue conceptuel, de définir des comportements génériques et réutilisables indépendamment des spécificités de l'agent, tout ceci dans un cadre méthodologique suffisamment précis pour orienter la conception.

Cela suppose de sortir du codage de l'agent ce qui peut être décrit de façon générique dans son comportement, comme le déplacement ou les interactions qu'il peut effectuer ou subir, et de ne lui laisser que des spécificités de perception et d'action, voire éventuellement un moteur de raisonnement. Quelle que soit la simulation, un comportement tel que *manger* correspond toujours à la même fonction ; il se décrira toujours avec les mêmes motivations (la faim) et les mêmes conditions d'exécution (posséder un objet mangeable). En revanche les détails de réalisation effective de l'interaction dépendent des particularités physiques de l'agent (ses caractéristiques élémentaires). Il en va de même pour le déplacement : quand on souhaite exprimer un comportement on peut être amené à représenter la nécessité d'effectuer tel ou tel déplacement (pour faire X il faut aller de A à B), mais on s'attache moins souvent à la façon de le réaliser (que ce soit avec un A* ou avec une exploration aléatoire).

Pour aller dans ce sens, nous proposons dans cet article de réifier la notion d'interaction indépendamment de la notion d'agent et de la notion de comportement. Cela amène à lister séparément d'une part, les activités qui peuvent être effectuées au cours d'une simulation, et d'autre part, les agents qui pourront les effectuer. Dans le modèle que nous proposons, les interactions sont des ensembles d'actions élémentaires soumises à des conditions d'exécution, les agents des entités plus ou moins complexes susceptibles d'effectuer ou de subir des interactions.
Nous montrons également comment la notion de comportement devient une simple conséquence des capacités d'interaction données aux agents, chaque interaction générique se trouvant réalisée selon des modalités qui dépendent des capacités élémentaires de perception, de cognition et d'action de chacun des agents.
Cette approche est particulièrement adaptée à des simulations nécessitant un nombre d'agents ou de sorte d'agents considérablement plus élevé que d'ordinaire (i.e. de l'ordre de plusieurs milliers ou dizaines de milliers), dans des domaines où les entités n'ont pas tant d'importance en tant qu'individus que par les relations fonctionnelles qu'elles entretiennent les unes avec les autres : c'est le cas des transports, de la biologie, des marchés, opposés à l'éthologie, l'IA, la théorie des jeux, etc.
Enfin, nous décrivons la méthodologie IODA[1] qui fournit des procédés empiriques d'analyse « centrée interactions » pour la conception de simulations s'appuyant sur notre modèle. Nous préconisons notamment traduire le comportement désiré sous forme de matrices dynamiques d'interactions entre les entités impliquées, avant de définir en détail la structure des agents et leurs capacités d'interactions.

2 La notion d'interaction

On peut mentionner des tentatives pour représenter formellement des abstractions dans les systèmes multi-agents, par exemple la notion de rôle qui a été introduite pour donner une réalité opératoire aux organisations [6]. Dans un tel cas, on s'attache en fait à dissocier les entités qui sont impliquées dans un système des relations organisationnelles qu'elles entretiennent les unes avec les autres, de façon à rendre ces dernières explicites et manipulables.
De la même manière, la notion d'interaction a fait l'objet d'explicitations à visées méthodologiques, comme par exemple dans l'approche Voyelles [4], sans toutefois conduire à une réalité informatique autonome. Ainsi, les interactions entre agents sont prises en compte lors de la modélisation, mais finissent par être codées dans un comportement d'agent *centré sur l'agent*. Dans le modèle que nous proposons, nous cherchons au contraire à donner un poids opérationnel égal aux entités et aux activités auxquelles elles prennent part.
Aussi, nous définissons les interactions comme des ensembles de primitives de base (les actions) qui impliquent simultanément plusieurs agents et qui constituent un bloc sémantique dans une simulation donnée [5]. Par exemple *manger* ou *ouvrir* ne sont pas de simples actions atomiques, mais correspondent à des ensembles structurés d'actions mettant en jeu deux agents différents et qui ne peuvent être effectuées qu'à certaines conditions, peu dépendantes des spécificités des agents concernés.
Ainsi formulée, cette notion d'interaction apparaît complètement dissociée des agents qui les utiliseront. C'est en effet une notion suffisamment générale pour pouvoir

[1] pour *Interaction-Oriented Design of Agent simulations*

être réutilisée dans différentes simulations et appliquée à des agents sources ou cibles différents : ainsi, l'interaction *ouvrir* a évidemment la même sémantique, et répond à la même fonctionnalité, qu'il s'agisse d'une simulation d'évacuation de bâtiment en cas d'urgence ou d'une chasse au trésor dans un jeu vidéo ; par ailleurs, en termes de prérequis et de conséquences sur l'état du monde, *ouvrir* se décrit de la même façon qu'on ouvre une porte, un coffre, ou une boîte de conserves. Cette dissociation est aussi un moyen très pratique pour les concepteurs de tester par la suite les conséquences comportementales de l'affectation d'une interaction à tel ou tel agent (c'est particulièrement le cas dans les jeux vidéo ou les mondes simulés).

Concevoir une simulation consiste donc d'abord à établir les primitives de base qui pourront être utilisées, à ensuite les agréger dans des interactions, et enfin affecter celles-ci aux agents.

2.1 Les primitives de base d'une simulation.

Elles fixent le niveau de granularité le plus petit qui puisse être représenté (en termes spatiaux : maillage, en termes temporels : pas de temps, et en termes comportementaux : briques de base des comportements).

Parmi ces primitives, on peut d'abord répertorier des *opérations de perception*, destinées à examiner l'état de l'environnement, les communications en provenance d'autres agents, les stimuli internes à l'agent (changement d'état), et le cas échéant les croyances ou buts. Nous ne faisons en effet aucune hypothèse particulière sur les capacités cognitives des agents auxquels peut s'appliquer notre modèle d'interaction, de sorte que la notion de « perception » peut aussi bien désigner un stimulus endogène ou exogène que l'aboutissement d'un processus de raisonnement sophistiqué.

La deuxième catégorie de primitives sert à se déplacer ou agir dans l'environnement, sur l'état de l'agent, ou sur d'autres agents, en incluant leur destruction ou la création de nouvelles entités. Ces *primitives d'action* peuvent elles aussi désigner aussi bien des réflexes simples produits en réponse à des stimulations réactives, que des actes issus d'un processus de planification ou d'une sélection d'action complexe.

Une interaction peut alors être définie comme *une séquence d'actions primitives, s'appliquant à plusieurs agents, déclenchées par des perceptions spécifiques et soumises à certaines conditions d'exécution*. Ces perceptions et ces actions primitives peuvent être réalisées selon des modalités variables par les agents, mais leur enchaînement logique est décrit de façon générale par l'interaction (cf. fig. 1).

Nous décrivons au § 2.4 comment ces interactions peuvent être affectées aux agents ; après quoi, la première condition pour qu'une interaction puisse avoir lieu entre deux agents, est que l'un d'entre eux soit capable *d'effectuer* cette interaction et que l'autre soit en mesure de *la subir*. À cela s'ajoute l'évaluation du déclencheur et des conditions.

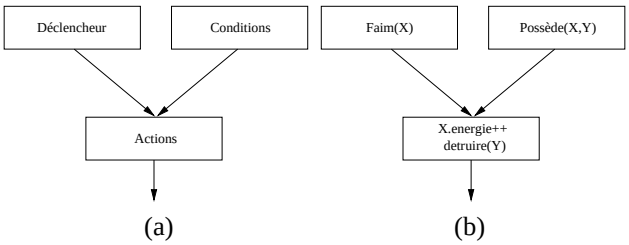

FIG. 1 – (a) Représentation générale d'une interaction. L'interaction est constituée d'un ensemble de perceptions (déclencheur d'une part, conditions d'exécution d'autre part) qui, lorsqu'elles sont réunies, permettent la réalisation d'une séquence d'actions. (b) Exemple : l'interaction *manger* se décrit de façon générale à partir d'une motivation interne (la faim) qui sert de déclencheur, de conditions d'exécution (posséder l'objet à manger) et de la séquence d'actions résultant de l'interaction : augmentation de l'énergie de l'agent source (celui qui effectue *manger*) et destruction de l'agent cible (celui qui subit *manger*).

2.2 Notions de déclencheur et de conditions.

Une interaction, en tant qu'expression abstraite d'un comportement, doit formuler les conditions portant sur la réalisation de la séquence d'actions primitives. Nous avons distingué deux composantes dans les tests exprimant les prérequis nécessaires à une interaction :

- le *déclencheur*, qui exprime une motivation (explicite ou implicite) à effectuer l'interaction : par exemple, le déclencheur de l'interaction *manger* est « avoir faim ». Ce déclencheur représente donc une forme de but : on mange pour éliminer la sensation de faim. Plus généralement, on peut s'attendre à ce que le déclencher exprime un besoin fonctionnel et que l'état du monde décrit par le déclencheur soit inversé ou corrigé par l'exécution de la séquence d'actions de l'interaction.
- la *condition* proprement dite, qui exprime les pré-requis matériels ou logiques pour pouvoir effectuer la séquence d'action : pour *manger*, il faut « posséder quelque chose de mangeable ». Sinon, il sera matériellement impossible de réaliser les primitives. Les conditions portent donc sur la possibilité physique ou logique d'effectuer la séquence d'actions.

2.3 Source et cible(s)

Une interaction, en principe, est effectuée par une source sur une cible. La source est un agent qui *peut effectuer* l'interaction, et la cible un agent qui *peut la subir*. Le moteur de simulation (cf. § 3.1) consiste essentiellement à scruter, pour chaque agent susceptible d'être la source d'une interaction, si des cibles potentielles sont atteignables (dans un certain rayon d'action), et choisir l'interaction de priorité la plus élevée pouvant être exécutée.

L'interaction ci-dessous, extraite d'une simulation des mécanismes de régulation génétique, exprime la traduction d'un gène. Pour que le gène subisse l'interaction *traduire*,

il faut qu'il rencontre une enzyme pouvant effectuer cette même interaction. Ils deviennent alors respectivement cible et source de l'interaction dont le code est donné ci-après. L'interaction ne peut effectivement avoir lieu (autrement dit, le code des actions ne peut être exécuté) que si la partie condition et la partie déclencheur sont vraies. La condition décrit le fait que la transcription ne peut avoir lieu que dans certains états de l'ADN ; quant au déclencheur, il effectue un tirage aléatoire selon le degré d'activité des facteurs de transcription pour déterminer si la traduction doit avoir lieu.

```
TraduireGene    :
      Condition(source,      cible)   :
            return   (cible.etat    != CONDENSÉ)
      Declencheur(source,       cible)   :
            if  (cible.activé      == ACTIVE)
              return   (random   < probaActivé)
            if(cible.activé      == NEUTRE)
              return   (random   < probaNeutre)
            if(cible.activé      == INHIBE)
              return   (random   < probaInhibé)
      Actions(source,      cible)   :
            creerNouvelleProteine(cible.proteineCodée)
```

Ce modèle général permet de plus de concevoir des interactions sans cibles définies : elles sont réalisées alors par un seul agent, la source, soit pour agir sur lui-même comme cible (changement d'état), soit pour agir sur l'environnement.

IODA permet également de spécifier des interactions s'appliquant à plusieurs cibles simultanément (toujours à partir d'une source unique) pour représenter des activités nécessitant une coordination entre agents. IODA peut donc exprimer des interactions entre une cible et une source, ainsi qu'entre une cible et plusieurs sources, le cas symétrique impliquant plusieurs sources et une cible pouvant se ramener au précédent en exprimant l'interaction à la voix passive. Par exemple, pour faire transporter une table par plusieurs agents, on n'utilisera pas l'interaction *transporter* qui requerrait plusieurs sources, mais plutôt l'interaction *être transporté* qui, elle, ne fait appel qu'à une source. La seule situation que IODA ne modélise pas est donc l'interaction simultanée entre plusieurs sources et plusieurs cibles (dont nous n'avons pas rencontré d'exemple...).

Il est à noter que cette représentation n'impose en rien de faire un choix sur l'axe réactif/cognitif, puisqu'une même interaction générique peut se traduire par des conditions et des actions soit réactives, soit cognitives selon les agents qui les exécutent. En revanche, selon la place accordée respectivement au déclencheur et aux conditions dans l'écriture des interactions, on peut concevoir une simulation orientée soit, à un extrême, par un jeu de fonctionnalités à assurer (prépondérance des déclencheurs qui vont mettre en avant des buts implicites ou explicites), soit à l'opposé par les seules caractéristiques structurelles des entités (prépondérance des réponses mécaniques déterminées par les possibilités d'exécution des actions).

2.4 L'affectation des interactions aux agents

Une fois définies les interactions susceptibles d'être réalisées au cours d'une simulation, il est en général facile de déterminer quels agents seront cibles ou sources. Il reste néanmoins deux points à préciser :

– La condition de distance entre la source et la cible pour que l'interaction puisse se produire. En effet, les agents n'interagissent potentiellement qu'avec des agents suffisamment « proches », qu'il s'agisse d'une distance spatiale dans l'environnement ou d'une mesure de proximité dans un espace d'états.
– La priorité que prend une interaction donnée lorsqu'elle est affectée à un agent donné, par rapport aux autres interactions qu'il est susceptible d'effectuer. Pour qu'un comportement rationnel puisse résulter des interactions entre agents, il faut en effet hiérarchiser ces interactions les unes par rapport aux autres, et ce d'une façon qui dépend assez étroitement des caractéristiques fonctionnelles des agents sources de cette interaction.

Il convient de souligner que ces caractéristiques ne peuvent en aucun cas relever uniquement soit de la définition des interactions, soit des propriétés des agents. Par exemple, l'interaction *manger* telle que définie dans la figure 1 doit être réalisée avec une distance source-cible nulle pour un individu normalement constitué, mais peut être effectuée à une distance de trois mètres pour un caméléon ! De même, elle sera certainement plus prioritaire que la reproduction chez un être humain que chez l'éphémère.

C'est lors de cette phase d'affectation des interactions génériques à des agents concrets, et lors de la définition des priorités et des gardes de distance associées à chaque interaction pour un agent donné, que l'on peut affiner les comportements produits au cours de la simulation. Au reste, rien n'exige que cette affectation reste inchangée au cours du déroulement de la simulation.

3 Le comportement d'un agent

Dans le modèle centré interactions, le comportement d'un agent, vu comme l'ensemble des actions qu'il accomplit effectivement au cours d'une simulation, est la résultante des interactions qui sont choisies. Pour cela, nous utilisons un moteur de sélection d'action « réactif », c'est-à-dire procédant mécaniquement à partir des interactions réalisables et de leurs priorités, sans chercher à satisfaire un objectif fonctionnel ou structurel global au sein du système. L'intérêt d'avoir séparé clairement les interactions des agents, mais aussi les interactions et les agents du moteur de sélection d'action, apparaît alors selon deux points de vue : d'une part en termes conceptuels (par rapport au domaine dans lequel s'inscrit la simulation), d'autre part en termes de génie logiciel et de réutilisabilité des interactions.

3.1 Un moteur de simulation réactif

Une fois définies les interactions, celles-ci vont être soumises à un processus général réalisant le mécanisme de sélection d'action.

Le principe que nous avons retenu est très simple : à chaque pas de temps, lorsque deux agents se rencontrent (au sens des contraintes de distance mentionnées précédemment),

l'un pouvant effectuer une interaction et l'autre pouvant la subir, l'interaction devient candidate, ce qui se résume avec la règle ci-dessous :
$I(x, y) \in$ *Interactions* peut avoir lieu si $\exists\ x, y \in$ *Agents* tels que *peut-effectuer*$(x, I) \wedge$ *peut-subir*(y, I)
Les prédicats *peut-effectuer* et *peut-subir* sont immédiatement donnés par les listes d'interactions dont les agents x et y peuvent être source ou cible.

Concrètement, un cycle de simulation consiste donc à demander à chaque agent d'essayer d'effectuer une interaction. Ce choix se déroule de la manière suivante (voir figure 2) :
– L'agent étudie d'abord la liste des interactions qu'il peut effectuer ; pour chacune d'entre elles il recherche des cibles potentielles (c'est-à-dire des agents pouvant subir ces interactions) respectant la garde de distance pour le déclenchement de l'interaction.
– Les interactions candidates sont rangées par ordre de priorité ; l'agent va alors évaluer pour chacune son déclencheur et ses conditions d'exécution, et choisir de réaliser la première interaction pour laquelle les deux seront vérifiés. Cela provoque la réalisation de la séquence d'actions.

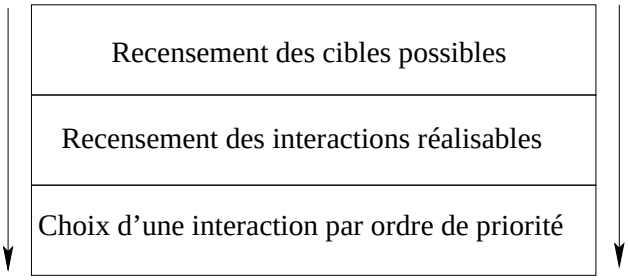

FIG. 2 – Mécanisme de sélection d'une interaction pour chaque agent à chaque pas de simulation

Si un agent n'est pas parvenu à exécuter une interaction, il reste susceptible d'en subir une. Un moteur correct pour ce modèle doit bien évidemment être équitable au sens où un agent pouvant effectuer une interaction ne doit jamais rester indéfiniment non sélectionné. En général nous travaillons par cycle d'évaluation de l'ensemble des agents, de sorte qu'un agent ne puisse effectuer ou subir qu'une seule interaction par cycle.

3.2 Structure des agents

En règle générale, les agents et leurs primitives constituent les aspects les plus dépendants du domaine lors de la réalisation d'une simulation. Mais dans notre approche, la donnée des interactions et du moteur réduit les agents à une importance moindre que dans les approches SMA classiques.
Dans notre modèle, un agent est une entité :
– dotée d'un état
– disposant de primitives de perception et d'action
– possédant des listes d'interactions pouvant être effectuées (avec leur priorité et gardes de distance) ou subies.

Ces spécifications simples suffisent à intégrer n'importe quel agent à notre modèle de simulation centré interactions ; elles ne préjugent en rien du degré de cognition de l'agent lui-même, de sorte que rien n'empêche le concepteur de choisir arbitrairement sa position sur l'axe réactif-cognitif.
Dans tous les cas, le moteur de sélection d'action que nous utilisons reste réactif : les interactions ne sont pas choisies en fonction par exemple d'un chaînage pour satisfaire un but, mais uniquement en fonction des possibilités de réalisation (existence d'une source et d'une cible, puis vérification du déclencheur et des conditions). Le processus de sélection des interactions est le même que l'agent ait des primitives de perception et d'action elles-mêmes réactives, ou que la validité des conditions et des déclencheurs fassent l'objet d'une preuve logique dans un modèle du monde symbolique.

3.3 Le point de vue conceptuel (représentation des connaissances d'un domaine)

Avoir « sorti » des agents le comportement qu'ils sont susceptibles d'adopter durant une simulation présente un premier intérêt en ce qui concerne la représentation des connaissances du domaine auquel s'applique la simulation. En effet, lorsque l'on cherche à utiliser les approches centrées individus dans des domaines autres que l'éthologie et la sociologie (par exemple la chimie, la biologie, la physique, etc.), la notion même de comportement perd de sa pertinence. Une même entité peut adopter au cours de son activité des comportements fort différents (par exemple, une enzyme peut selon sa forme effectuer des fonctions cellulaires sans rapport les unes avec les autres), et inversement des entités structurellement assez éloignées (avec des propriétés physico-chimiques assez différentes) peuvent avoir des activités fonctionnellement équivalentes (une protéine ou une molécule d'ARN peuvent jouer un rôle régulateur dans l'expression d'un gène).
Avoir réifié en tant que telle la notion d'interaction permet donc de définir d'une part des ontologies d'entités intervenant dans un processus, et d'autre part des ontologies de fonctionnalités assumées par ces entités. La relation entre ces deux ensembles n'est en règle générale ni bijective, ni figée au cours d'une simulation.
Représenter les modifications radicales de comportement qui peuvent affecter certaines entités revient donc en fait à modifier les listes d'interactions que cette entité peut effectuer ou subir. C'est une approche plus souple que lorsqu'il faut « adapter » le comportement d'un agent par une sélection d'action sophistiquée.

3.4 Le point de vue Génie Logiciel

Le second avantage de notre modèle consiste simplement en la réutilisation, d'une simulation à une autre, des bibliothèques d'interactions définies de façon générique. Si

toutes les interactions d'une simulation ne sont évidemment pas utilisées dans une autre, il n'en reste pas moins que, pour des domaines suffisamment proches, les interactions communes se formulent souvent de façon identique. Il ne reste alors qu'à récrire, pour une simulation spécifique, que les agents propres au nouveau contexte, avec leurs primitives de perception et d'action propres au nouveau contexte.

Nous ne prétendons pas ici qu'avec notre approche centrée interactions il est possible de réaliser des simulations impossibles à faire avec les autres plateformes. La thèse que nous soutenons est que, du point de vue Génie Logiciel, la conception de simulations et la réutilisation de bibliothèques d'interactions est grandement facilitée par cette approche. Dans le type de problème ciblé ici, les plateformes les plus utilisées sont sans aucun doute Swarm, Netlogo et Madkit. Dans ces trois plateformes il n'y a pas de modèle d'agent, ni en termes d'architecture, ni en termes de primitives de perception et d'action. L'agent est une coquille vide dans lequel le concepteur peut/doit mettre l'ensemble du code nécessaire au comportement. Il en résulte dans le code de l'agent un mélange entre ce qui relève de l'action sur l'environnement (positionnement et déplacement en particulier), des interactions avec les autres agents, et de la technique de sélection d'action. Tout ceci étant entremêlé, il est alors très difficile d'ajouter ou supprimer une possibilité comportementale et bien évidemment de réutiliser du code dans une simulation similaire.

```
to go  ;; forever  button
  ask turtles  [ go-turtles  ]
  diffuse  chemical  (diffusion-rate  / 100)
  ask edge-patches
    [ set chemical  0 ] ;; prevent  wrapping
  ask patches  [ go-patches  ]
  do-plotting
  set clock  (clock  + 1)
end

to go-turtles  ;; turtle  procedure
  if (who  < clock)  ;; delay
  [ ifelse  carrying-food?
    [set color  orange  + 1 return-to-nest  ]
    [set color  red  look-for-food  ]
  ]
end

to go-patches  ;; patch  procedure
  set chemical  (chemical  *(100-evaporation-rate)/100)
  update-display  ;; Refresh  the  Display
end
```

L'exemple ci-dessus, tiré de la simulation *Ants*, l'une des plus célèbres de la bibliothèque d'exemples NetLogo, est assez représentative. En dépit de la simplicité de ce langage procédural, on peut voir qu'il y a un mélange complet entre des notions spécifiques à cette simulation (par exemple les phéromones représentées par la variable `chemical` et susceptibles de diffuser), et un « bricolage » lié à l'absence de séparation entre modèle et vue (par exemple le fait de mettre à zéro cette variable `chemical` sur les bords, simplement parce que l'environnement Netlogo est systématiquement torique !). On se doute que dès que le modèle du comportement des fourmis est modifié, déterminer quelles procédures doivent être modifiées d'une part, et garantir qu'on n'introduit pas d'artefacts d'autre part, constituent clairement un casse-tête, de sorte qu'en général il est nécessaire de réécrire complètement le code.

Dans les plateformes ouvertes disposant de toutes les avancées modernes du Génie Logiciel, il n'en reste pas moins que la réutilisabilité porte sur les agents vus comme un tout (en tant que classes d'un langage objet par exemple), avec toute la dépendance vis-à-vis du domaine dont leur comportement peut être imprégné. Au contraire, l'extirpation de ce comportement hors de l'agent, sous la forme d'interactions formulées de façon abstraite, permet de gagner en généricité. On peut alors réutiliser d'une part les agents en tant qu'entités (ou structures), d'autre part les interactions en tant qu'expression d'une fonction.

4 Représentation de connaissances adaptée au contexte large échelle

La simulation centrée-individu est initialement limitée à quelques dizaines ou centaines d'agents [7, 8]. Or, de nombreux domaines présentent des besoins croissants à des échelles bien plus importantes, en ce qui concerne à la fois le nombre total d'agents (plusieurs milliers à plusieurs centaines de milliers) et le nombre de classes d'agents (plusieurs centaines). C'est le cas par exemple du cinéma (scènes de bataille du film *Le Seigneur des Anneaux*) qui utilise des agents pour remplacer des figurants dans des combats réalistes, des jeux vidéo dont les joueurs non humains doivent faire preuve de plus en plus d'autonomie et de rationalité pour ne pas donner l'impression de « tricher », des transports (INRETS), et enfin de la biologie.
Ce dernier domaine met particulièrement en lumière les contraintes propres aux simulations à large échelle. Il s'agit d'abord de *contraintes de nombre* : une seule bactérie compte environ 4 000 espèces moléculaires différentes, pour la plupart susceptibles d'interagir les uns avec les autres, et de l'ordre de 10^{10} molécules au total. On comprend qu'il est en soi problématique de réaliser une simulation dans laquelle un agent représenterait une molécule. Aussi, la plupart des modèles de biologie (étude des réseaux de régulation par exemple) sont-ils basés sur des équations aux dérivées partielles portant sur les concentrations de quelques espèces moléculaires. Or, cette approche écarte d'emblée les spécificités spatiales des phénomènes biologiques, et ne permet pas d'aborder les situations dans lesquelles quelques molécules seulement sont impliquées (jouant par exemple un rôle de signalisation [9]). Il faut pour cela tenir compte d'une autre caractéristique des phénomènes « large échelle », à savoir la coexistence au sein des mêmes processus *d'échelles* fort éloignées [10] : échelles dans les ordres de grandeur des agents (des millions de protéines de structure contre quelques enzymes signalétiques), échelles spatiales (l'environnement local des cellules et les modifications de conformation de l'ADN), et

échelles de temps (quelques secondes à quelques heures).

Enfin, les simulations par agents étant apparues dans des domaines comme l'éthologie, l'économie, la sociologie, il est évident que bon nombre de concepts de ces disciplines, et tout particulièrement la notion de *comportement* propre aux acteurs intervenant dans les phénomènes visés, ont été importés sans questionnement épistémologique approfondi. Or, le passage à des domaines à large échelle relativise fortement la notion même *d'individu*, et par suite celle de *comportement individuel*. En physique, en biologie, en écologie ou en dynamique des marchés, le comportement de chaque entité participant au processus étudié importe nettement moins que les *interactions* qui s'y déroulent. Dans le cas de la physique statistique par exemple, cela est tellement vrai que c'est en intégrant les interactions (chocs) entre particules qu'on détermine la loi des gaz parfaits. L'individualité des particules n'a que peu d'intérêt. Dans la majorité de ces domaines, nous sommes loin de cet extrême, car les agents ont une histoire qui les rend non interchangeables, mais leur activité en tant qu'individu n'a guère de sens.

5 Vers une analyse centrée interactions

5.1 Esquisse de méthodologie : IODA

Attaquer un problème de simulation suppose d'identifier à la fois les entités qui, selon le modèle du domaine ciblé, sont supposées interagir les unes avec les autres pour produire le phénomène étudié, et ces interactions elles-mêmes. Dans un modèle de simulation centré agents, l'identification se focalise sur les entités, que l'on dote ensuite de comportements destinés à produire les interactions voulues. Les fonctions abstraites associées aux interactions sont ainsi perdues et « engluées » dans la spécificité des agents.

Nous suggérons au contraire de mener de front l'analyse des agents et des interactions, de façon à garder une vue abstraite des fonctionnalités assurées par les agents.

Notre méthodologie actuelle, IODA (*Interaction-Oriented Design of Agent simulations*), propose ainsi trois étapes pour la conception d'une simulation centrée interactions :

1. Identifier les interactions (fonctionnalités abstraites, processus élémentaires). Cela conduit à dresser une matrice entre sources et cibles potentielles (cf. tab. 1, et tab. 4 pour un exemple détaillé) dans laquelle des interactions génériques apparaissent clairement. Cette matrice est susceptible d'évoluer au cours de la simulation, selon la manière dont les agents sont affectés par les interactions.
2. Identifier les caractéristiques des agents concernés (structure et capacités élémentaires de perception et d'action), et leur affecter les interactions qu'ils peuvent effectuer ou subir, en spécifiant en outre la priorité relative des interactions pouvant être effectuées et leur garde de distance. Cela conduit à un tableau de synthèse (cf. tab. 2, et tab. 5 pour un exemple détaillé).
3. Déterminer enfin la dynamique du système, c'est-à-dire la façon dont, au fil des interactions, évoluent les caractéristiques des agents, y compris leurs possibilités d'interaction. Cela peut amener à construire un diagramme de transitions entre divers états du système, correspondant chacun à des matrices d'interactions différentes. Ainsi, dans l'exemple détaillé donné au tableau 3, on peut distinguer dans la simulation « AOE » une alternance entre un état « paix » avec une activité de patrouille et une économie classique d'une part, et un état « guerre » avec une activité militaire ciblée et une économie d'urgence d'autre part. Ces changements d'état s'accompagnent de « masques » qui modifient l'affectation des interactions aux diverses classes d'agents.

Afin d'aider au suivi de ces différentes étapes, nous cherchons dans la phase d'analyse-conception à définir des tableaux-types qui permettent avant tout de répondre aux questions suivantes : d'une part, quelles sont les interactions, avec quelles sources et quelles cibles ; d'autre part, quels sont les agents, comment peuvent-ils interagir ? ; enfin, quels sont les dynamiques d'affectation des interactions aux différents agents ?

Sources \ Cibles	Ag_1	Ag_2	...	Ag_n
Ag_1				
...	...	...	...	...
Ag_n				

TAB. 1 – Ce premier tableau fournit la matrice des interactions réalisables dans la simulation étudiée. Une interaction I placée sur la ligne i et la colonne j peut être effectuée par l'agent Ag_i et subie par Ag_j.

agent	carac	p_effectuer	priorité	dist	p_subir
...	...	...	...	...	...

TAB. 2 – Ce deuxième tableau donne pour chaque type d'agents de la simulation étudiée, ses caractéristiques structurelles et la liste des interactions pouvant être effectuées (avec, le cas échéant, leur priorité et leur garde de distance), et subies.

5.2 Extension du modèle : les déplacements

L'approche que nous avons décrite dans cet article a été en grande partie implémentée et testée sur une plateforme dédiée à la simulation large échelle, SimuLE, dont plusieurs démonstrations sont disponibles sur le site de l'équipe. Cet

États	Agent	peut effectuer	peut subir
Paix	Soldat	¬*Combattre*	¬*Combattre*
Paix	SoldatEnnemi	¬*Combattre*	¬*Combattre*
Paix	SoldatEnnemi	¬*Détruire*	
Paix	Forum		¬*Détruire*
Guerre	Forum	¬*CréerPaysan*	

TAB. 3 – Le dernier tableau recense les éventuelles modifications d'affectation des interactions aux agents, par rapport à la matrice d'interactions initiale, en fonction de la dynamique souhaitée pour la simulation. Dans cet exemple, appliqué à la simulation « Age of Empires » (AOE), dans l'état « Paix » les soldats ne peuvent pas combattre : on supprime donc de la liste de ce qu'il peuvent effectuer l'interaction *Combattre* (présente au contraire dans l'état « Guerre »). De même pour économiser des ressources le Forum cesse de produire des Paysans en temps de guerre.

outil est actuellement utilisé pour des applications en biologie cellulaire, notamment pour évaluer les différentes hypothèses liées à la transcription génétique. Ces simulation nécessitent des milliers d'agents de familles différentes avec de nombreuses interactions entre chacune d'entre elles ; SimuLE est capable actuellement de faire interagir jusqu'à 50 000 individus simultanément en temps réel à l'écran avec de très nombreuses interactions. SimuLE possède aussi un dispositif original d'abonnement à des indicateurs statistiques pour chaque simulation et est capable de fournir un rendu aussi bien 2D que 3D du modèle simulé (cf. fig. 3). Pour des raisons d'optimisation et de rapidité, la distance est dans la version actuelle, propre à chaque agent (halo de perception des cibles potentielles) et non liée à la connexion interaction-agent comme il a été décrit précédemment. L'une des prochaines améliorations de cette réalisation consistera bien évidemment à régler ce point.

Nous envisageons plusieurs extensions pour accroître la distinction conceptuelle et logicielle entre les agents et la représentation de leur activité. En particulier, le déplacement des agents, et plus généralement les lois physiques de l'environnement, auxquelles sont soumis les agents, ne sont pas à proprement parler des « interactions », dans la mesure où ces lois s'exercent en permanence sur tous les agents, et viennent s'ajouter aux interactions que les agents ont les uns avec les autres.

Le comportement d'un agent résulte du fait qu'il est soumis à des lois physiques exercées par l'environnement et qu'il interagit avec d'autres agents. De même que nous avons séparé logiciellement et conceptuellement les interactions des agents, nous projetons de distinguer les lois physiques (contraintes sur le déplacement par exemple) des agents et/ou des interactions dans lesquelles elles sont actuellement représentées. Ainsi, nous disposerons de bibliothèques de déplacements génériques réutilisables, auxquelles les entités simulées pourront être soumises indépendamment de leurs activités comportementales.

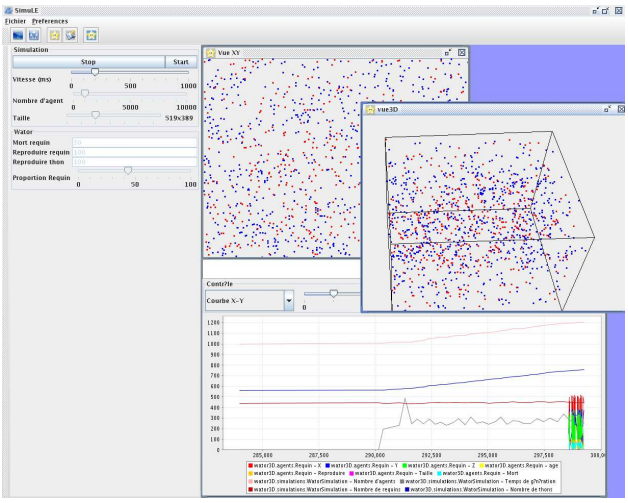

FIG. 3 – Une capture d'écran de SimuLE, avec plusieurs vues de la simulation en cours et un suivi de statistiques.

6 Conclusion

Dans cet article nous avons présenté un modèle formel pour l'interaction entre agents situés pour la simulation. Ce modèle part du constat que lors de la phase d'analyse d'un problème, il n'y a aucune raison de lier les différentes interactions possibles avec les agents qui seront mis en présence, contrairement à ce qu'offrent les plateformes actuelles. Cette séparation des concepts permet notamment une réutilisabilité accrue des interactions créées et permet sans aucun doute de concevoir dans des contextes similaires des bibliothèques d'interactions réutilisables. La validation de cet argument comprendrait l'évaluation du temps nécessaire pour modifier une simulation, comparé à un codage centré sur l'agent. Ce type de chronométrage n'est cependant pas simple à mettre en œuvre, tout comme il est malaisé de comparer sur ce point les paradigmes de programmation objet et impératif par exemple.

Nous avons montré qu'une conception rigoureuse passait par différentes étapes commençant par l'identification et la création des interactions possibles ainsi que des agents qui seront impliqués, se poursuivant par l'affectation de ces interactions aux agents avec la prise en compte de la distance et enfin de la gestion des priorités. Le modèle d'agent que nous préconisons s'appuie sur deux propriétés fondamentales nommées *peut-subir* et *peut-effectuer* qui contiennent chacune une liste d'interactions caractérisant le comportement de l'agent. Nous avons enfin décrit précisément comment concevoir le moteur réactif permettant de prendre en charge ce modèle.

Ce modèle formel a été implémenté dans la plateforme SimuLE de l'équipe SMAC/LIFL qui permet notamment de faire interagir plus de 50 000 agents en temps réel dans un environnement 2D ou 3D. La méthodologie IODA

Sources \ Cibles	Forum	Borne	Mine	Soldat	Soldat ennemi	Paysan	(aucune)
Forum				DonnerRole	Combattre	DonnerRole	CreerSoldat CreerPaysan NeRienFaire
Borne				Router			
Mine							
Soldat					Combattre		DevenirChef DeplChef Deplacement
Soldat ennemi	Détruire			Combattre		Combattre	DevenirChef DeplChef Deplacement
Paysan	PrevenirMineEpuisee DeposeRessource		MineEpuisee PrendreRessource				Deplacement

TAB. 4 – Matrice des interactions réalisables dans la simulation « Age of Empires » (AOE) testée sur la plateforme SimuLE. Les agents manipulés dans cette simulation sont : le forum (qui crée des paysans et des soldats, et où l'on peut déposer des ressources), les mines contenant des ressources, les paysans qui exploitent les mines, les soldats défendant les paysans et le forum, des soldats ennemis, et des bornes entre lesquelles les soldats patrouillent. Cette matrice indique quelles interactions sont susceptibles d'être effectuées et subies par chaque classe d'agents. Pour modéliser des interactions sur plusieurs cibles, on peut en outre indiquer une cardinalité à côté de l'interaction.

agents	caractéristiques	peut effectuer	priorité	distance	peut subir
AgentAOE Mouvement	direction pointsVie camp role positionForum				
Soldat ennemi	chef pointsAttaque	DevenirChef Combattre Detruire DeplacementChef Deplacement	4 3 2 1 0	 $d \leq 5$ $d \leq 2$	Combattre
Soldat	chef pointsAttaque	DevenirChef Combattre DeplacementChef Deplacement	3 2 1 0	 $d \leq 4$	Combattre DonnerRoleSoldat Router
Paysan	ressources ressourcesMaximum typeRessources mineEpuisee provenance	MineEpuisee PrevenirMineEpuisee DeposerRessource PrendreRessource Deplacement	2 2 1 1 0		DonnerRolePaysan Combattre
Forum	mines minesEpuisees bornes timerCreation pointsVie ressources	Combattre DonnerRoleSoldat DonnerRolePaysan CreerSoldat CreerPaysan NeRienFaire	4 3 3 2 1 0	$d \leq 6$	Detruire DeposerRessource PrevenirMineEpuisee Detruire
Mine	ressources typeRessources				PrendreRessource MineEpuisee
Borne	direction	Router	0		
Obstacle					

TAB. 5 – Liste donnant pour chaque type d'agents de la simulation « Age of Empires » (AOE), ses caractéristiques et la liste des interactions pouvant être effectuées (avec, le cas échéant, leur priorité et leur garde de distance), et subies.

a été utilisée et affinée au cours de la mise au point des expériences menées sur cette plateforme. Actuellement, SimuLE est notamment utilisée pour tester différentes hypothèses dans plusieurs problèmes de biologie cellulaire, domaine nécessitant une plateforme large échelle.

En ce qui concerne les limites de notre approche, IODA prend clairement son essor avec des simulations large échelle, tant en nombre d'interactions qu'en nombre d'agents. Un écosystème trivial avec un type de proie et un type de prédateur, et une seule interaction *manger* sera sans doute plus rapidement écrit en NetLogo.

De même que nous avons dissocié les agents de leur comportement, nous envisageons de distinguer les lois physiques des agents et des interactions, afin de construire des bibliothèques de déplacement génériques et réutilisables. Ces lois sont en effet l'expression de contraintes environnementales et s'appliquent indépendamment de la nature des agents ou de leur comportement. Nous disposerons donc à terme d'une triple ontologie d'agents, d'interactions et de lois environnementales qui permettra de concevoir plus efficacement des simulations dans des contextes changeants.

Remerciements

Ce travail est cofinancé par le contrat de plan Etat-Région et les fonds européens FEDER.

Références

[1] R. Burkhart. « The Swarm Multi-Agent Simulation System ». *Object-Oriented Programming Systems : Langages and Applications (OOPSLA)*, Workshop on The Object Engine, 1994.

[2] O. Gutknecht, J. Ferber, F. Michel, « Integrating tools and infrastructures for generic multi-agent systems ». In : *Proceedings of the fifth international conference on Autonomous agents (AA'01)*, ACM Press p. 441–448, 2001.

[3] U. Wilensky. NetLogo. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL, 1999. http ://ccl.northwestern.edu/netlogo/ .

[4] Y. Demazeau, « From Interactions to Collective Behaviour in Agent-Based Systems ». In *Proceedings of the 1st European Conference on Cognitive Science*, Saint-Malo, 1995.

[5] P. Mathieu, S. Picault et J.-C. Routier. « Simulation de comportements pour agents rationnels situés ». *Actes de la conférence Modèles Formels pour l'Interaction (MFI'03)*, p. 277–282, 2003.

[6] J. Ferber et O. Gutknecht, « A meta-model for analysis and design of organizations in multi-agent systems ». *Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS'98)*, p. 128–137, 1998.

[7] M. Resnick, *Turtles, termites and traffic jams. Explorations in massively parallel microworlds*, MIT Press, 1997.

[8] J.M. Epstein et R. Axtell, *Growing Artificial Societies. Social Science from the Bottom Up*, MIT Press, 1996.

[9] R.C. Paton, M. Fisher et K. Matsuno. « Intracellular signalling proteins as 'smart' agents in parallel distributed processes ». *Biosystems*, 1999.

[10] S. Leibler, L.H. Hartwell, J.J. Hopfield et A.W. Murray. « From molecular to modular cell biology », *Nature* vol. 402, déc. 1999.