Une approche méthodologique pour la conception et l'analyse de simulateur multi-agents

Fabien Michel

Laboratoire d'Informatique, Robotique et Micro-électronique de Montpellier.
C.N.R.S. – Université Montpellier II, 161 rue Ada 34392 Montpellier Cedex 5 - France
http://www.lirmm.fr, fmichel@lirmm.fr

Résumé

Dans cet article, nous présentons une approche méthodologique originale de la conception de simulateur multi-agents basée sur un modèle organisationnel. Ce modèle, appelé Aalaadin, permet une description simple de structures d'organisation et d'interactions dans les systèmes multi-agents (SMA). Nous montrons que l'utilisation de ce modèle permet d'envisager simplement et efficacement l'analyse du problème fondamental du scheduling dans les simulations multi-agents. Ce problème consiste à assurer que les agents d'une simulation sont soumis à une même loi temporelle et qu'ils agissent « tous en même temps ». Lié aux difficultés de modéliser la simultanéité entre agents, ce problème fondamental a très peu été étudié car difficile à expliciter. Nous décrivons ici le fonctionnement d'un agent de la plate-forme MadKit nommé scheduler qui permet de surmonter ces difficultés d'analyse et de conception. Son principe est de décomposer le problème du scheduling global en autant de sous-problèmes que nécessaire. Ainsi nous montrons comment son fonctionnement permet de construire des politiques d'exécution toujours plus raffinées qui restent compréhensibles grâce d'une part à la localisation des problèmes et, d'autre part, à l'explicitation de la structure organisationnelle du simulateur.

Mots clés

Systèmes Multi-Agents, simulation, scheduling, organisation.

1 Introduction

Bien que point crucial dans la simulation multi-agents, très peu de recherches portent sur le problème de la modélisation de la concurrence des interactions dans la simulation. Comme l'écrit Ferber : « le problème est patent : alors que nous supposons en permanence que les actions sont effectuées en parallèle, nous ne disposons pas de formalisme adéquat pour décrire facilement des actions simultanées et donc représenter des actions collectives. »[1].

Cette problématique, qu'on désignera ici par « problème du scheduling » ou autrement dit « politique d'exécution des agents », n'a en effet quasiment aucun support méthodologique alors que paradoxalement ce problème est un passage obligé lors de l'implémentation d'un simulateur multi-agents.

En effet, le paradigme multi-agents est fondé implicitement sur la composition de comportements individuels simultanés[2]. De plus les architectures multiprocesseurs n'apportent pas de solution évidente : sans synchronisation les agents évoluent au rythme de la complexité de leur architecture interne, ce qui ne permet pas de contrôler la cohérence de la dynamique globale du système[3].

Ainsi toutes les solutions envisagées pour simuler la concurrence des entités d'un système sont génératrices de biais dans le processus de simulation. Autrement dit l'implémentation de ce point précis a un impact direct sur l'évolution du modèle multi-agents considéré alors qu'on souhaiterait sa neutralité.

Malgré cela, très peu de réflexions portent sur l'analyse de ces biais. Sur ce point, la réflexion de Drogoul en 1993 [4] concernant les systèmes multi-agents (SMA) en général est encore valable aujourd'hui : « la littérature consacrée aux systèmes collectifs ne s'est jamais vraiment penchée sur la question de leur efficacité, évitant sans doute, ce qui est compréhensible, de remettre prématurément en cause de manière expérimentale des principes mal définis théoriquement. ».

Sans apporter ici de réponse au problème de la modélisation de la simultanéité, nous pensons que le problème de la conception et de l'analyse du scheduling doit prendre une place prépondérante lors de la conception d'une simulation de SMA. Il est donc impératif de se donner des moyens méthodologiques et pratiques qui permettent une analyse simple et, surtout, explicite de ce problème.

Il ne s'agit pas ici de vérifier une hypothétique correspondance avec la réalité mais bien de mieux connaître l'outil d'expérimentation et ses implications dans le résultat final, notamment afin de pouvoir évaluer les approximations et les erreurs liées directement à celui-ci.

Dans ce cadre, notre hypothèse de travail est que tout simulateur de SMA peut lui-même être exprimé sous la forme d'un SMA et que, par nature, ce système définit une structure organisationnelle sous-jacente. L'utilisation explicite de cette structure doit apporter une solution aux problèmes que nous avons soulevés tant au niveau de la description que de la mise en place des mécanismes de scheduling.

Nous verrons comment cette approche peut facilement être mise en œuvre dans le cadre du méta-modèle organisationnel Aalaadin [5]. La force de ce modèle, basé sur trois concepts centraux -l'agent, le groupe, le rôle- est de permettre une description simple et puissante des structures d'organisation et d'interactions dans un SMA sans a priori sur l'architecture interne des agents.

Ainsi son utilisation nous permettra de définir simplement la dynamique d'un SMA en décomposant le problème de la synchronisation globale en plusieurs sous-problèmes définis au niveau des différents groupes et rôles émergeants de la structure du modèle de simulation dans son ensemble.

Nous montrerons comment cette méthode nous permet d'implémenter simplement des politiques d'exécution très différentes pouvant coexister à l'intérieur d'une seule et même simulation.

Dans la première partie de cet article nous précisons les enjeux de cette problématique et nous ferons une synthèse des techniques de scheduling actuelles ainsi que des problèmes

qu'elles soulèvent. Dans la section suivante nous décrivons une approche organisationnelle du problème. Nous présentons succinctement le modèle Aalaadin et nous décrirons une implémentation correspondant à notre approche. La quatrième section expose un exemple d'application utilisant un simulateur développé sur ces principes et nous terminons par quelques perspectives.

2 Problématique et état de l'art

2.1 Le scheduling : au cœur du processus de simulation

Si de façon classique une simulation consiste à expérimenter des modèles donnés sous la forme de relations mathématiques entre des variables représentant des grandeurs physiques mesurables dans la réalité, au contraire la simulation multi-agents se propose de représenter directement les individus, leurs comportements et leurs interactions.

Enumérer tous les travaux basés sur la simulation multi-agents serait très fastidieux. On peut cependant noter que la plupart ont pour objet soit la simulation effective d'un modèle multi-agents spécifique : une colonie de fourmis [6], des robots footballeurs [7], soit le développement d'une plate-forme plus ou moins générique liée à un domaine d'application particulier : la gestion de ressources renouvelables (Cormas [8], Echo [9]), la robotique mobile (MissionLab [10]), l'éthologie (LiveWorld [11]).

L'étude de la problématique liée à leur implémentation est en général centrée autour d'une ou plusieurs des notions suivantes :

- Le type des agents simulés : communicant, réactif, cognitif, situé...
- L'environnement dans lequel évoluent les agents : discret, continu, 2D, 3D...
- La nature des interactions : coordination, négociation, perceptions et actions sur l'environnement, actes de langage…

Cette approche, qu'on peut qualifier d'orientée « modèle multi-agents » est justifiée dans la mesure où les domaines d'application sont extrêmement variés. Les modèles considérés peuvent être de nature très différente et nécessiter une analyse poussée.

Une telle approche a l'inconvénient de ne pas faire apparaître de façon explicite les problèmes liés à l'implémentation de la simulation du temps. Alors qu'étant donné l'étendue des domaines d'application, il est clair que chaque modèle ne nécessitera pas le même niveau de granularité temporelle : du temps réel pour la robotique mobile à la simulation de plusieurs années pour des problèmes écologiques.

Par ailleurs, une simulation multi-agents suppose qu'on a implémenté un mécanisme permettant de synchroniser les actions des agents. Σ définissant l'ensemble des états possibles du système, toutes simulations multi-agents est basée sur l'hypothèse que l'évolution de l'instant t à l'instant t+dt de l'état du monde $\sigma \in \Sigma$ résulte de la composition des actions A1(t), A2(t)… An(t) produites par les agents à t De façon simplifiée, il s'agit de construire une fonction du temps, Dynamique D, telle que :

$$D : \Sigma \rightarrow \Sigma$$

$$\sigma(t+dt) = D (\Pi Ai(t), \sigma(t))$$

Le symbole Π est ici utilisé pour désigner l'opérateur de composition des actions. Il définit de quelle manière les actions produites à un instant t doivent être sommées afin de calculer leurs conséquences sur l'état du monde.

Sans entrer dans le détail, il est facile de mesurer la difficulté de conceptualiser une telle opération étant donné la multitude et la nature des concepts qui peuvent se cacher derrière le mot action (mouvement, prise de décision, modification de l'environnement). De plus la simultanéité des actions est extrêmement difficile à modéliser et la théorie manque beaucoup sur ce point. Ainsi tous les concepteurs de simulation multi-agents sont amenés à faire un choix personnel en la matière.

Et ce choix est difficile : un modèle multi-agents particulier ne définit pas une technique a priori (puisque la problématique de ces techniques est liée à la simultanéité et non au modèle). Un même modèle peut donc être implémenté, à ce niveau là, de multiples façons.

A partir de là il est simple de voir qu'un seul modèle théorique peut donner des résultats différents suivant la technique de scheduling utilisée.

C'est pourquoi au contraire des mathématiques (dont la validité et la constance ne dépendent pas d'un modèle) dans les simulations numériques, la méthode de scheduling utilisée pour implémenter un modèle multi-agents est indissociable des résultats obtenus.

De façon classique l'expérimentation sert à évaluer la qualité d'un modèle. Elle doit permettre de le valider ou de le remettre en cause en vue d'une modification. Nous pensons que dans le cadre de la simulation multi-agents, il est impératif d'avoir la même démarche vis-à-vis de la politique d'exécution employée.

2.2 Les techniques de scheduling actuelles

Dans cette section nous présentons les trois grands types de synchronisation utilisés dans les plates-formes actuelles de simulation: à « pas de temps constant », avec état tampon (parfois désigné par « double buffer ») et par événements.

2.2.1 Les simulations à «pas de temps constant »

Cette méthode consiste à activer les agents (et éventuellement les objets de la simulation) de façon séquentielle. L'activation de l'ensemble du système correspond alors à un pas de temps pour la simulation.

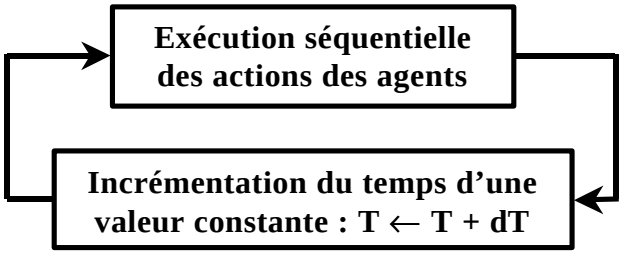

Figure 2.1. Principe d'une simulation à pas de temps constant

Cette technique a l'avantage d'éviter la présence de conflits dans l'accès aux variables de l'environnement et de garantir que tous les agents agissent une fois par cycle. Par ailleurs, c'est de loin la technique la plus utilisée du fait de sa simplicité. Les plates-formes StarLogo[12] et Manta[6] sont des exemples de l'utilisation de cette technique.

Notre objet n'est pas ici d'énumérer tous les biais engendrés par ce type de fonctionnement. Nous allons cependant illustrer notre propos à l'aide de deux exemples simples.

Notre premier exemple décrit un modèle de simulation connu sous le nom « proies / prédateurs ». On pose qu'une proie (un triangle) est capturée lorsqu'elle est entourée de quatre prédateurs (les ronds). On trouvera une description plus exhaustive de ce problème, ainsi que des résultats expérimentaux dans [1]. On voit sur la figure 2.2 que l'ordre d'exécution des agents (ici représenté par un numéro) peut changer l'issue d'une même situation. Ici la vie ou la mort de la proie.

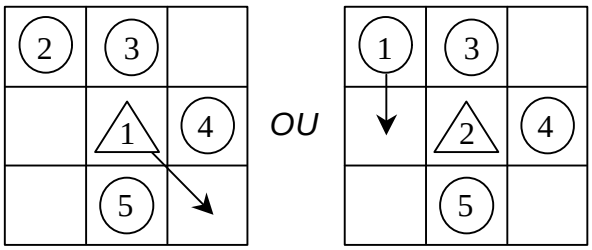

Figure 2.2. La survie de la proie dépend de sa place dans la liste d'exécution

Notre deuxième exemple met l'accent sur le problème du choix de la granularité des actions pour un pas de temps. Cet exemple est tiré du modèle de simulation de la plate-forme StarLogo [12]. Une tortue (les dénominations des agents dans ce modèle) peut effectuer à chaque pas de temps une action primitive. Ici la primitive considérée est *fd n*. Elle signifie que la tortue avance de n cases.

La figure 2.3 montre que plusieurs tortues peuvent par exemple avoir des trajectoires qui se croisent sans être à aucun instant de la simulation sur la même case, la vitesse des déplacements n'ayant pas de lien direct avec le temps[1].

[1] Les tortues 1, 2 et 3 avancent l'une après l'autre de deux cases. Puis viens le tour de la tortue 4 qui avance d'une case. On voit par ailleurs que si la tortue 4 s'était déplacée en première, les autres l'auraient trouvée sur leur trajectoire.

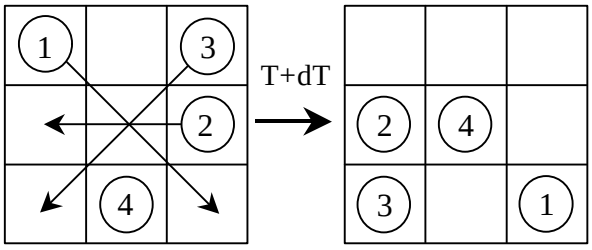

Figure 2.3. Les tortues se sont croisées sans être à aucun moment sur la même case

2.2.2 Simulation avec état tampon ou « double buffer »

Extension de la précédente, cette technique vise à apporter une solution au problème de la simultanéité.

L'objectif est de faire en sorte qu'à un instant t tous les agents aient la même perception de l'état du monde $\sigma(t)$.

Pour cela les actions des agents se font sur des variables tampons et le monde n'est pas directement modifié. Une fois tous les agents exécutés, on fait la synthèse de l'ensemble des actions pour calculer le nouvel état du monde.

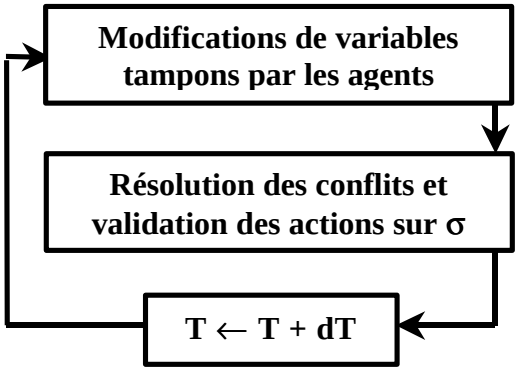

Figure 2.4. Un modèle de simulation avec état tampon

Les simulations de type jeu de la vie sont basées sur ce principe : l'état du monde est mis à jour une fois que toutes les cellules ont calculé leur état suivant. Les plates-formes Cormas [8] et LiveWorld [11] proposent ce mode de fonctionnement.

L'utilisation de cette méthode entraîne cependant directement un nouveau problème. Un conflit apparaît lorsque deux agents ou plus spécifient de nouvelles valeurs différentes pour une même variable. Il faut donc éventuellement résoudre les conflits aboutissant à une incohérence de l'état du monde.

Par ailleurs, Travers qui utilise cette méthode dans sa plate-forme LiveWorld, y voit lui-même un autre sérieux désavantage: si un agent modifie une variable, il ne peut pas s'en servir pour un nouveau calcul sans risquer d'obtenir un résultat incohérent, la valeur de la variable n'étant validée qu'à la fin d'un cycle.

Sans entrer dans le détail des inconvénients liés aux résolutions de conflits, on peut simplement remarquer qu'on en revient, d'une façon ou d'une autre, à instaurer un mécanisme de priorité entre les agents lors de la résolution d'un conflit.

Ainsi, même si cette méthode se fonde sur une analyse de la situation plutôt que sur une liste d'ordonnancement prédéfinie, on peut facilement retrouver des situations semblables à celles de nos exemples précédents.

2.2.3 Simulation par événements

Plutôt que de synchroniser tous les agents à un instant t, l'idée de cette méthode est d'exhiber explicitement un ordre chronologique entre les actions des agents. Le principe consiste à déterminer a priori ou en cours de simulation les événements futurs, leur date et leur nature.

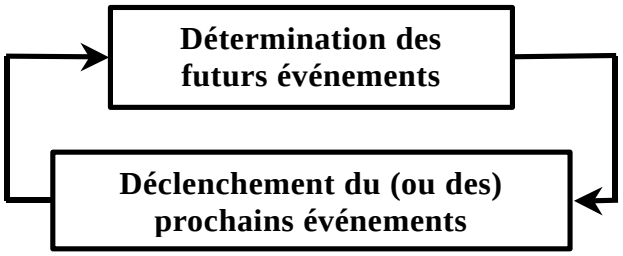

Figure 2.5. Principe d'une simulation par événements

Magnin utilise par exemple dans son simulateur Sieme [7] un ensemble de règles environnementales permettant de déterminer au fur et à mesure des interactions les événements à déclencher: *Si <condition événement> alors <déclencher événement>*. Ainsi un principe de causalité entre événements est au cœur de ses préoccupations.

Au contraire dans la plate-forme Swarm [13] la chronologie des interactions est déterminée a priori. Un swarm (un SMA) est défini comme une collection structurée d'objets à laquelle est associée un ordonnancement des activités de ces objets. C'est cet enchaînement qui définit la logique de la dynamique du système[2].

Bien que ce type de simulation semble loin des précédents, il n'en reste pas moins qu'on retrouve les mêmes genres de problèmes. En effet, lorsque plusieurs événements sont concurrents les difficultés à représenter la simultanéité restent les mêmes. Magnin en convient d'ailleurs lui-même [7] : « Si dans le modèle Sieme les règles gèrent les interactions entre les entités du système, aucune prise en compte générique des actions n'est proposée. En fait, actuellement une action se limite à l'exécution du code ayant un accès direct aux attributs de l'entité qui agit. Or le résultat d'une action n'est pas uniquement lié à l'entité agissante. ».

2.3 Première conclusion

Nous pensons que le problème de la clarté des simulateurs actuels ne tient pas particulièrement aux biais engendrés par une technique précise de scheduling. Comme nous l'avons dit : il faut faire un choix. Nous pensons plutôt que le simple fait de faire en la matière un choix unilatéral est une erreur.

Il faut remarquer que le fonctionnement d'une simulation est toujours basé sur une unique politique d'exécution. Ainsi tous les agents de la simulation sont en général soumis à un seul et même principe de simulation. A partir de là, la tâche peut s'avérer excessivement complexe dès lors qu'on s'attaque à un modèle comportant des agents très différents. En effet la granularité des actions (mouvement, vitesse, etc.) et la sémantique des interactions (collisions, coordination, etc.) peuvent être très différentes.

De plus il n'est pas rare que les processus qui accompagnent la simulation (affichages, analyse statistique, etc.) soient eux aussi gérés sur le même principe. Dans un système événementiel par exemple, l'affichage correspond à un événement particulier qu'il faut gérer de la même façon que les autres.

Ainsi le fait que tout le système soit soumis à la même méthode de scheduling contribue à rendre obscur le fonctionnement du simulateur, ce qui a pour conséquence de rendre son analyse difficile et de limiter ses possibilités d'extension à d'autres agents.

3 Le simulateur comme un SMA

3.1 Organiser pour régner

Notre idée est simple et d'ailleurs elle n'est pas nouvelle dans le principe. Il s'agit de diviser le problème du scheduling global en autant de sous problèmes que nécessaire. En effet il suffit de remarquer que – quelle que soit la méthode – synchroniser tous les agents n'a plus vraiment de sens à partir du moment où on considère plusieurs types d'agents, chacun de ces types pouvant nécessiter l'emploi d'une technique de scheduling différente.

Il est plus simple de se concentrer sur un groupe d'agents dont la synchronisation est cruciale pour le processus de simulation. En décomposant le problème on élabore des groupes formés d'agents dont la synchronisation des actions est considérée comme nécessaire. Chaque groupe peut alors être traité indépendamment de façon à identifier un protocole de scheduling adapté à la situation. Ces groupes définissent alors naturellement une structure organisationnelle dont la nature reflète de façon explicite la logique du simulateur.

3.2 Le modèle Aalaadin et la plate-forme MadKit

Nous présentons ici très rapidement le modèle de description organisationnelle Aalaadin et la plate-forme MadKit basée sur celui-ci. Ce modèle a été décrit en détail dans [5]. Sa

[2] Par exemple un swarm peut être, comme dans [13], un ensemble de lapins et de coyotes associé à la suite d'actions suivante : les lapins mangent des carottes puis fuient les prédateurs et les coyotes mangent ensuite les lapins. Un pas de temps correspond alors à l'exécution de toutes les actions dans l'ordre défini. Chaque action peut elle-même être décomposée suivant une liste d'événements discrets.

sémantique opérationnelle basée sur le π-calcul a été proposée dans [14]. On trouvera par ailleurs les perspectives méthodologiques associées dans [15].

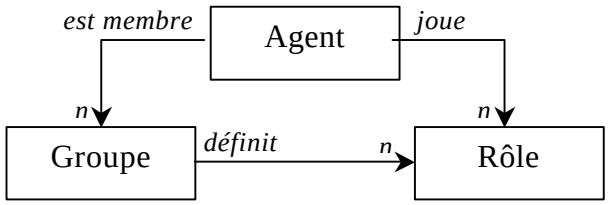

Figure 3.1 Modèle organisationnel

3.2.1 Agent

Quasiment aucune contrainte n'est posée sur l'architecture interne ou sur le modèle de l'agent. *L'agent* est simplement décrit comme une entité autonome communicante qui joue des *rôles* au sein de différents *groupes*. Cette très faible sémantique est volontaire. Le but est de laisser toute liberté au concepteur pour choisir l'architecture appropriée à ses besoins.

3.2.2 Groupe

Le groupe est la notion primitive de regroupement d'agents. Chaque agent peut être membre d'un ou plusieurs groupes. D'une façon un peu simpliste, un groupe peut être vu comme un moyen d'identifier par regroupement un ensemble d'agents. Plus classiquement, associé au rôle, il définira la structuration organisationnelle d'un SMA usuel. Les différents groupes peuvent par ailleurs se recouper librement.

3.2.3 Rôle

Le rôle est la représentation abstraite d'une fonction, d'un service ou d'une identification d'un agent au sein d'un groupe particulier. Chaque agent peut avoir plusieurs rôles, un même rôle peut être tenu par plusieurs agents, et les rôles sont locaux aux groupes.

3.2.4 La plate-forme MadKit

La structure organisationnelle que nous venons de décrire est implémentée au cœur de la plate-forme MadKit, tant pour fournir un modèle organisationnel aux SMA exécutés que pour le fonctionnement interne du système.

Par ailleurs, à l'instar des autres plates-formes multi-agents, celle-ci n'a pas été conçue pour un domaine d'application précis mais pour la conception de SMA en général. Les avantages d'un tel processus méthodologique sont par ailleurs discutés dans [16].

De plus MadKit intègre deux principes supplémentaires : une architecture à micro-noyau[3] et l'agentification systématique des services.

3.3 L'agent Scheduler de MadKit

Notre idée est d'avoir transposé à la simulation les principes de conception de la plate-forme MadKit.

Nous avons donc implémenté un micro-noyau dédié à la simulation appelé moteur synchrone composé de deux agents qui offre des services liés à la conception de simulation.

L'un de ces deux agents[4] permet donc la conception de techniques hétérogènes de scheduling : l'agent Scheduler.

Son rôle consiste à manipuler **des** politiques d'exécution sur lesquelles aucune contrainte n'est posée. Ainsi cet agent est associé à un objet outil générique appelé Activateur.

Dans sa forme la plus simple un Activateur est simplement le moyen pour l'agent Scheduler d'identifier un ensemble d'agents étant donné un groupe et un rôle. Par exemple, le Scheduler peut créer un Activateur sur le rôle *agent* dans le groupe *simulation*, ou encore *afficheur* dans le groupe *représentation graphique*.

Ainsi, le principe est de spécialiser des sous-classes d'Activateur de façon à définir une technique de scheduling particulière qui pourra ensuite être appliquée ponctuellement, par le scheduler, à différents groupes d'agents.

L'avantage réside dans le fait qu'un même agent Scheduler peut créer autant

[3] Le terme micro-noyau est utilisé ici intentionnellement comme référence au modèle des systèmes d'exploitation à micro-noyau dont le principe fondateur est la création d'un ensemble de fonctions clé permettant le développement efficace de services système.

[4] Le deuxième agent, *Watcher*, est utilisé selon les mêmes principes que le scheduler en ce qui concerne les mécanismes d'observation de la simulation.

d'activateurs que nécessaire. Sa tâche se résume simplement à ordonner l'exécution de ses activateurs pour définir le fonctionnement du simulateur dans son ensemble. La figure 3.2 décrit par exemple un simulateur comportant un agent Scheduler *S* qui utilise deux types d'activateur (deux politiques d'exécution différentes) sur trois groupes d'agents différents. De plus cette figure montre qu'un agent peut appartenir à plusieurs groupes simultanément.

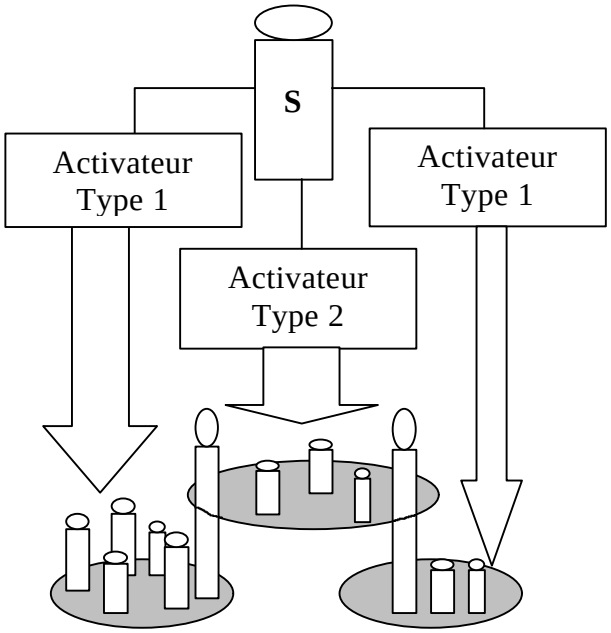

Figure 3.2 Un simulateur « organisationnel »

Le principal avantage de la décomposition du scheduling global réside dans la possibilité de modifier ou de remplacer un activateur sans avoir à toucher la structure globale : on a découplé les problèmes de gestion de la simulation (Scheduler) de celui de la synchronisation des agents (Activateur), le Scheduler n'étant pas responsable de la qualité d'un activateur. Un agent Scheduler peut par ailleurs, en tant que simple agent faire l'objet d'un contrôle supérieur.

3.4 Méthodologie de conception d'un simulateur organisationnel

Ainsi nous proposons une méthodologie de conception de simulateur multi-agents en trois étapes :

1. Exprimer la structure organisationnelle du simulateur sous la forme de groupes et de rôles.
2. Elaborer les types d'activateurs nécessaires au simulateur (c'est-à-dire des techniques de scheduling locales).
3. Définir le fonctionnement global du simulateur en ordonnant l'exécution des activateurs.

L'énoncé de la troisième étape peut laisser croire que nous nous trouvons face à la même problématique que celle liée à l'ordonnancement des agents. En fait, il s'agit bien de deux niveaux d'analyse distincts.

En effet un activateur, lié à un groupe, définit une technique de scheduling particulière (à pas de temps constant, par événements, etc.) pour des agents dont la synchronisation est considérée comme nécessaire.

Au contraire, l'ordre d'exécution des activateurs décrit simplement la logique du simulateur, c'est-à-dire l'ordre dans lequel les différents groupes clés (agents simulés, affichage, observation, etc.) interviennent dans le processus de simulation.

La figure 3.3 illustre cet aspect de notre approche. Elle décrit un exemple d'ordonnancement de quatre activateurs différents A1, A2, A3 et A4. Chaque activateur est lié à un ensemble d'agents identifié par un groupe et un rôle. Les politiques de scheduling utilisées par les différents activateurs sont indépendantes. A1 peut par exemple utiliser une méthode à pas de temps constant alors que A2 définit un principe événementiel.

Cependant il est évident que l'ordre d'exécution des activateurs peut influer sur les résultats. Ainsi si on inverse A1 et A2, on peut obtenir des résultats différents étant donné que ces activateurs manipulent des agents (agent type1 et agent type2) qui peuvent modifier le monde. Néanmoins il ne s'agit pas ici

d'un problème de synchronisation : la décomposition en deux activateurs implique qu'on suppose que les différents groupes d'agents ne doivent pas intervenir en même temps.

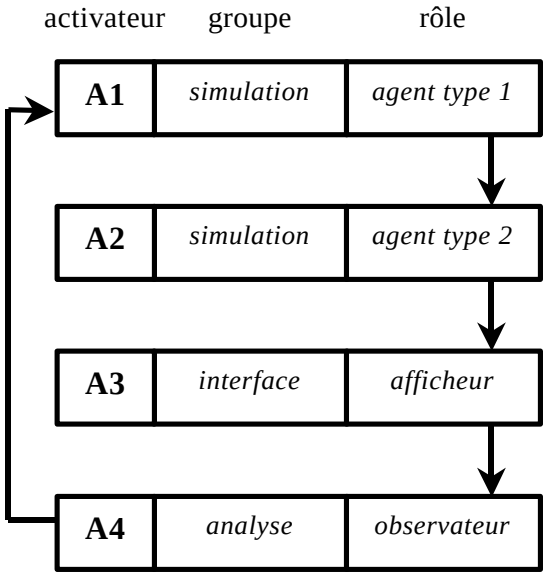

Figure 3.3 exemple d'application du principe de simulation par activateurs

Par contre il est tout à fait envisageable de considérer, après analyse des résultats, que la synchronisation de deux groupes est finalement nécessaire. Il est alors très intéressant de pouvoir changer localement le fonctionnement du simulateur. On peut par exemple imaginer la fusion de A1 et A2 en un seul activateur plus adéquat, A3 et A4 restants valides.

L'interchangeabilité des activateurs, obtenue grâce à la vue organisationnelle, trouve alors tout son sens : elle permet de constituer rapidement plusieurs simulations basées sur différentes techniques de scheduling sans avoir à recoder les agents de la simulation. Par ailleurs, rien n'est supposé sur l'architecture des agents simulés. Ainsi, il est possible d'utiliser des agents très différents de façon transparente : il suffit pour cela d'affecter à un agent le groupe et le rôle correspondants pour que celui-ci intègre le processus de simulation.

Nous soutenons qu'une telle méthodologie permet de construire une politique d'exécution très complexe au niveau global qui reste compréhensible grâce à la décomposition du problème. De plus nous pensons que la possibilité de pouvoir modifier **localement** le fonctionnement du simulateur permet d'envisager la réalisation de simulations toujours plus complexes comportant un grand nombre d'agents fortement hétérogènes.

On notera également que l'approche organisationnelle a montré d'autres avantages dans le cadre de la simulation notamment au niveau de la modélisation de l'environnement [19].

4 Application

4.1 Le TurtleKit de MadKit

Le TurtleKit est une bibliothèque de simulation pour agents réactifs qui clone une partie des fonctionnalités de l'environnement StarLogo [12]. Ce modèle de simulation définit les rôles suivants : *turtle* (les agents de la simulation), *environment* (fonction de gestion du monde), *displayer* (fonction d'affichage du monde) et *observer* (pour les agents chargés de l'analyse de la simulation). Un Activateur spécifique, le *TurtleActivator*, a été implémenté pour gérer l'exécution des turtles.

Bien que le modèle de simulation considéré soit simpliste, l'utilisation du moteur synchrone montre déjà plusieurs avantages.

Premièrement, les agents considérés n'ont en rien besoin d'avoir la même architecture interne. Ainsi il est possible à un utilisateur d'utiliser ses propres agents dans le processus de simulation. Il lui suffit pour cela d'affecter les rôles correspondant au système de simulation considéré. Ce qui permet par exemple d'avoir plusieurs représentations graphiques simultanées gérées par différents agents jouant le rôle de *displayer*. Ces agents peuvent de plus être lancés en cours de simulation : le simple fait de jouer un rôle précis engendre la participation d'un agent au processus de simulation.

Deuxièmement on peut facilement envisager de changer localement la gestion du scheduling des turtles sans avoir à toucher une seule ligne de code dans les agents.

Troisièmement tous les agents de la simulation sont de vrais agents MadKit. Par conséquent ils possèdent exactement les mêmes

fonctionnalités que des agents non simulés (passage de message, vue organisationnelle, etc.).

Pour finir, il est tout à fait possible de lancer plusieurs simulations simultanément en créant plusieurs groupes différents pour chaque simulation sachant qu'un rôle est local à un groupe.

4.2 Conception et analyse d'une simulation test

Pour illustrer notre approche nous avons utilisé le TurtleKit pour réaliser une simulation basée sur un modèle de type proies/prédateurs (coyotes/lapins) à l'image de celui décrit dans [13]. Les proies peuvent adopter deux comportements : manger (*R_eat*) ou fuir les prédateurs (*Hide*). Trouver un lapin à dévorer (*C_eat*) est par contre le seul comportement des coyotes.

Les résultats d'une telle simulation résident dans la fluctuation du nombre d'individus de chaque espèce. De façon classique l'évaluation d'un modèle consiste à faire varier les paramètres de départ (ici le nombre de proies et de prédateurs).

En ce qui nous concerne, nous ne ferons varier que la technique de scheduling. Nous présenterons tout d'abord les différentes techniques employées puis nous discuterons ensuite les résultats obtenus.

La première technique utilisée est un calque de celle proposée dans [13] : les lapins mangent des carottes puis fuient les prédateurs et les coyotes mangent ensuite les lapins. (*R_eat* puis *Hide* et enfin *C_eat*). Cette vision du déroulement de la simulation fait clairement apparaître deux groupes, *lapins* et *coyotes*, pour lesquels on va définir un activateur pour chaque comportement (3 activateurs en tout). L'enchaînement de ces activateurs définit alors le fonctionnement du simulateur. Notons que dans ce cas de figure, ce fonctionnement est similaire à celui de la plate-forme Swarm.

La première variation qui vient à l'idée est de changer l'ordre des activateurs. Nous avons donc inversé les deux derniers activateurs pour obtenir la séquence suivante : *R_eat* puis *C_eat* et enfin *Hide*.

Ensuite, il nous a semblé judicieux de reconsidérer le fait que les agents soient activés de manière indépendante. Par ailleurs, il semble intéressant que les proies ne puissent adopter qu'un seul comportement par cycle, manger **ou** fuir, de façon à ne pas être « privilégiées » par rapport aux prédateurs. Nous avons donc utilisé un unique activateur (le *TurtleActivator* par défaut) lié à un groupe formé des deux espèces (tous les agents jouent le rôle *turtle* par défaut). Cet activateur permet à chaque agent de décider de son futur comportement de façon dynamique. Ainsi une proie aura un comportement de fuite tant qu'elle ne se sentira pas en sécurité.

Pour finir nous avons modifié le *TurtleActivator* de manière à ce que chaque agent enchaîne trois comportements successivement, autrement dit un agent est maintenant trois fois plus rapide. Cela de manière à évaluer l'impact de la granularité temporelle des actions.

Il nous faut préciser ici que chacune de ces configurations utilise strictement le même code pour tous les agents de la simulation. Ainsi, la seule différence entre chaque modèle de simulation se situent dans la technique de scheduling utilisée.

En ce qui nous concerne nous ne nous intéresserons pas à l'interprétation des résultats, nous ne ferons que les comparer entre eux pour évaluer l'impact de la technique de scheduling utilisée. La figure 4.1 est une copie d'écran qui montre l'évolution de la population des proies en fonction du nombre de cycle sur une vingtaine de tests pour chaque type de simulation (1, 2, 3 et 4).

Sur cette figure on constate que si les valeurs sont extrêmement proches pour un même type de simulation (ce qui paraît cohérent), seules les deux premières solutions donnent des résultats équivalents.

En revanche on observe des différences très nettes avec les deux dernières solutions. Les résultats sont mêmes extrêmement éloignés tant au niveau des valeurs que des pentes des courbes.

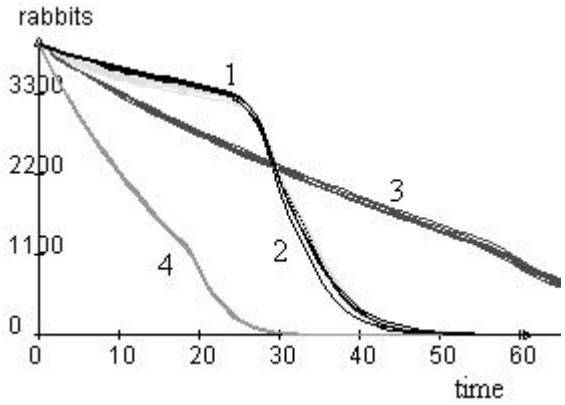

Figure 4.1 impact du scheduling sur des résultats de simulation

Malgré la simplicité de la méthode employée, on réalise ici l'impact que peut avoir une technique de scheduling sur le déroulement d'une simulation. C'est pourquoi il semble donc tout à fait intéressant de pouvoir concevoir et analyser plusieurs de ces techniques comme nous venons de le faire.

5 V Conclusion et perspectives

Dans cet article nous avons vu que le problème du scheduling est un point clé de la simulation multi-agents.

Nous avons défendu l'idée que seule une décomposition explicite du problème global de scheduling permet d'envisager l'édification de politiques complexes qui restent compréhensibles (donc analysables) grâce à une localisation des problèmes de synchronisation.

Le modèle Aalaadin, que nous avons présenté ici permet une décomposition logique du simulateur en utilisant la structure organisationnelle sous-jacente au SMA formé par tous les agents de la simulation.

Notre objectif à long terme est de proposer de nouvelles techniques de simulation qui permettront, par leur simplicité et leur interchangeabilité, d'expérimenter plus avant un modèle multi-agents dans le cadre de la simulation.

Il semble par exemple tout à fait pertinent d'imaginer expérimenter un même modèle suivant plusieurs politiques d'exécution comme nous l'avons fait dans la quatrième section.

Nous pensons que cela permettra de faire le premier pas vers une analyse plus approfondie et plus fructueuse des mécanismes de simulation multi-agents.

6 Références

[1] J. Ferber. Les systèmes multi-agents, vers une intelligence collective. Pages 175, 399-411, Interéditions, 1995.

[2] M. Resnick. New Paradigms for Computing, new Paradigms for Thinking. *Computers and Exploratory Learning, edited by A. diSessa, C. Hoyles, & R. Noss. Springer-Verlag*, 1995.

[3] M. Lhuillier. Objets et Agents pour systèmes d'information et de simulation. *Thèse de doctorat de l'université Paris VI*, pages 37-40, 1998.

[4] A. Drogoul. De la simulation multi-agents à la résolution de problème collectif. *Thèse de doctorat de l'université Paris VI*, page 145, 1993.

[5] J. Ferber et O. Gutknecht. A meta-model for the analysis and design of organizations in multi-agent systems. *Third International Conference on Multi-Agent Systems (ICMAS '98) Proceedings, IEEE Computer Society*, pages 128-135, 1998.

[6] A. Drogoul, B. Corbara et D.Fresneau. MANTA: New Experimental Results on the Emergence of (Artificial) Ant Societies. *Actes de Simulating Societies, Sienne*, 1993.

[7] L. Magnin. Modélisation et simulation de l'environnement dans les systèmes multi-agents : application aux robots footballeurs. *Thèse de doctorat, université Paris VI*, 1996.

[8] F. Bousquet, I. Bakam, H. Proton et C. Le Page. Cormas : common-pool ressources and multi-agent systems. *Lectures Notes in Artificial Intelligence 1416*, pages 826-837, 1997.

[9] P.T. Hraber, T. Jones et S. Forrest. The Ecology of Echo. *In* Artificial *Life 3*, pages 165-190, 1997.

[10] D. MacKenzie, R.C. Arkin et R. Cameron. Multiagent Mission Specification and Execution". *Autonomous Robots*, Vol. 4, No. 1, pages 29-52, Jan 1997.

[11] M.D. Travers**.** Programming with Agents: New metaphors for thinking about computation. *Thesis, Massachusetts Institute of Technology*, pages1996.

[12] M. Resnick. Learning about life. *Artificial Life*, vol. 1, no. 1-2, spring 1994.

[13] N. Minar, R. Burkhart, C. Langton et M. Askenazi. The Swarm simulation system : A toolkit for building multi-agent simulations. *Rapport n°96-06-042, SantaFe Institute*, 1996.

[14] J. Ferber et O. Gutknecht. Operationnal semantics of multi-agent organization. *Workshop on Agent Theories, Architectures and Languages (ATAL '99),*to be published in Intelligent Agents Series, Springer-Verlag, 1999.

[15] O. Gutknecht et J. Ferber. Vers une méthodologie organisationnelle pour les systèmes multi-agents ». GLEIZES M.-P, *Actes des journées Francophone en Intelligence Artificielle Distribuée et Systèmes Multi-Agents 1998*, Hermès, Nov. 1999

[16] O. Gutknecht, J. Ferber et E. Lieurain, Des modèles hétérogènes de simulation par systèmes multi-agents. *Colloque SMAGET 98, Systèmes multi-agents et Gestion de l'environnement et du territoire*, Cemagref, Clermont-ferrand, 6-8 octobre 1998 Cemagref Editions.