Simulation de comportements pour agents rationnels situés

P. Mathieu S. Picault JC. Routier
mathieu@lifl.fr picault@lifl.fr routier@lifl.fr

Laboratoire d'Informatique Fondamentale de Lille
Université des Sciences et Technologies de Lille
59655 Villeneuve d'Ascq cédex – FRANCE

Résumé :
Nos travaux visent à simuler des comportements vivants qui soient perçus comme vraissemblables (rationnels) par un observateur humain, par exemple la simulation du comportement des personnes en cas d'incendie dans un immeuble à évacuer ou la modélisation de personnages dans les jeux vidéo. Nous nous plaçons clairement ici dans le cadre du Génie Logiciel et de l'Intelligence Artificielle distribuée.
Actuellement, force est de constater qu'il n'existe pas d'AGL (Atelier de Génie Logiciel) de modélisation de comportements comme il peut exister des moteurs graphiques réutilisables dans différentes applications.
Nous souhaitons montrer dans cet article que du point de vue de la réutilisabilité et du génie logiciel il y a un manque à combler entre la génération de plans et les systèmes réactifs communément utilisés pour la simulation de comportements.

Mots-clés : Simulation, Interaction, Comportement, Agents Situés, Généricité

Abstract:
Our research aims at simulating life-like behaviors that can be seen as plausible (rational) by a human observer, e.g. simulating the behavior of people that have to clear a building during a fire, or the modelling of characters in video games. This work should clearly be seen in the context of Software Engineering and Distributed Artificial Intelligence.
At present, it is to notice that there is no Integrated Development Environment (IDE) dedicated to modelling of behaviors, as it exists graphical engines which can be reused in various applications.
In this paper, we would like to demonstrate that, from the viewpoints of reusability and software engineering, there is a gap to fill between plan generation on the one hand, and reactive systems often used for behavior simulation on the other hand.

Keywords: Simulation, Interaction, Behaviour, Situated Agents, Genericity

1 Introduction : la simulation informatique

L'informatique a modifié dans une très large mesure la manière dont les sciences expérimentales appréhendent leur objet, non pas tant en fournissant des outils de calcul puissants pour les sciences mathématisées (essentiellement la physique) qu'en provoquant graduellement, à travers la notion de *simulation*, une évolution de la façon de penser le rapport du modèle à l'expérience. En particulier, les concepts issus de la programmation par objets puis des systèmes multi-agents ont permis de donner une réalité informatique à la notion de *comportement*. Dans le même mouvement, l'informatique a produit également des programmes de simulation dont les objectifs ne sont pas liés à la recherche scientifique ou à la validation de modèles d'autres disciplines : c'est le cas par exemple des environnements de réalité virtuelle ou des jeux vidéo. Quelle que soit sa fonction, la simulation informatique ne fait cependant pas l'objet d'un traitement générique : il n'existe pas de "moteur de simulation de comportements" qui serait l'analogue pour l'IA des moteurs d'animation graphique. Or, si une généricité absolue n'est sans doute pas atteignable, nous souhaitons montrer ici, en nous appuyant sur une comparaison des diverses méthodes employées, que la simulation du *comportement d'agents rationnels situés* fait appel à un ensemble de caractéristiques opérationnelles clairement identifiables qui peuvent être rassemblées dans un modèle générique, dont nous proposons une réalisation. Nous nous plaçons donc ici dans le domaine de l'IA Distribuée [5] qui s'intéresse aux mécanismes d'interaction et d'organisation de groupes d'agents. Les agents *situés*, en particulier, ont pour caractéristique d'appartenir à un espace qui joue un rôle constitutif de leur comportement.

Nous défendons l'idée que, pour reproduire des comportements rationnels réalistes dans des agents situés, il est nécessaire de construire un modèle d'interaction spécifique, capable de tenir compte des caractéristiques propres à ce type d'agents. Un tel formalisme, centré sur les interactions, doit permettre de traiter de façon homogène des simulations de comportements destinées aussi bien à l'investigation scientifique qu'à des moteurs de jeux vidéo.

1.1 Qu'est-ce qu'une simulation ?

Dans la majorité des sciences expérimentales, la simulation consiste "à pratiquer des tests *sur un substitut* de la situation réelle [...] (le substitut étant supposé reproduire fidèlement les carac-

téristiques pertinentes essentielles de la situation à caractériser" [7]. Elle doit être mise en œuvre lorsque l'*expérience* normalement destinée à mettre à l'épreuve un modèle n'est pas réalisable. Il s'agit fréquemment de dispositifs physiques, comme les souffleries pour l'étude des caractéristiques aérodynamiques. Toutefois, comme la plupart des modèles physiques ont une base mathématique, le terme de "simulation" a rapidement été étendu à la résolution exacte ou approchée de modèles équationnels par des calculateurs électroniques. Actuellement, une "simulation", en informatique, désigne tout programme qui se substitue à une *expérience*. Deux grandes catégories d'applications sont donc visées : la réalisation d'expériences ou de simulations qui auraient été menées par des moyens physiques dans d'autres disciplines ou la production de situations artificielles.

1.2 Simuler des comportements rationnels : un besoin croissant

Un besoin émergent. La simulation de comportements rationnels a été très tôt l'un des objectifs phares de l'informatique, à travers le projet de l'Intelligence Artificielle : reproduire les compétences intellectuelles de l'être humain. Outre les difficultés de cette approche, abordée initialement sous un point de vue essentiellement logique et linguistique, il est apparu très vite que nombre d'*applications* de l'IA ne nécessitaient pas une intelligence anthropomorphique.

Cela n'est plus complètement vérifié. On recense des champs de recherche relativement nouveaux qui nécessitent une forme d'IA de type humain, non pas tant pour *résoudre des problèmes* spécialement complexes mais plutôt pour *interagir harmonieusement* avec des acteurs humains. Citons par exemple la robotique sociale [1], l'utilisation de la réalité virtuelle à des fins d'entraînement, ou, surtout, les jeux vidéo [6].

Un domaine difficile. Ces derniers constituent en particulier selon J. Laird l'application ultime pour l'IA de niveau humain [6] ; car les personnages des jeux vidéo doivent présenter une autonomie et un réalisme sans cesse croissants. Ils doivent être *vraisemblables*, donc leur comportement doit se conformer aux attentes rationnelles du ou des joueurs humains qui sont leurs partenaires ou leurs adversaires. Il leur faut également s'adapter à des situations nouvelles, acquérir des compétences supplémentaires au fil du jeu, etc. Enfin, des stratégies d'équipe leur sont souvent nécessaires.

Pour mener à bien des interactions de ce type, le comportement des agents doit pouvoir être perçu par un observateur humain comme "intelligent" ou "rationnel", c'est-à-dire qu'un être humain doit estimer que le comportement de l'agent vise effectivement à atteindre les buts qu'il s'est fixés d'une façon proche de celle qu'aurait produite une intentionnalité humaine [3]. Il ne s'agit pas, heureusement, d'un point de vue purement subjectif (qui risquerait de menacer la possibilité même d'implémenter des comportements "rationnels" de ce genre). Les sciences cognitives ont mis en lumière un certain nombre de contraintes (logiques, pragmatiques, sociales) qui contribuent, dans les situations d'interaction, à rendre les comportements humains mutuellement intelligibles [4, 2], de sorte que l'impression de "rationalité" que donne un comportement repose bien sur les techniques qui ont été mises en œuvre pour produire ce comportement.

Dans le cas des jeux vidéo, ces contraintes "cognitives" se doublent de contraintes "économiques" : la rapidité du développement, qui passe par la réutilisabilité de travaux antérieurs. Plus généralement, ce domaine combine des difficultés qu'on rencontre en IA classique (la représentation des connaissances), en IA distribuée (la coordination d'agents ayant des buts individuels souvent différents), et en Génie Logiciel (la réutilisabilité des outils conceptuels et logiciels).

L'approche générique de la simulation que nous défendons suppose d'être capable, au moyen d'un même formalisme, de produire des modèles informatiques utilisables aussi bien pour la simulation de phénomènes naturels que pour la production de comportements vraisemblables dans un monde artificiel. Dans un cas comme dans l'autre, diverses approches existent déjà. Nous les décrivons sommairement dans ce qui suit pour les comparer et les mettre en regard de nos attentes.

2 Vers un formalisme générique

2.1 Caractéristiques souhaitées

Comme cela a été mentionné précédemment, notre approche suppose de pouvoir représenter

à la fois le niveau individuel et le niveau collectif. Si l'on souhaite de plus pouvoir réutiliser le même formalisme d'un domaine à un autre, et d'une simulation à une autre pour un même domaine d'application, il est nécessaire de séparer le plus possible les connaissances spécifiques à un domaine du moteur d'interactions, et d'expliciter autant que possible ces connaissances. Il faut noter qu'aucune des approches classiques ne permet de remplir tous les critères simultanément. Notre objectif est de fournir un cadre **uniforme** et **générique** pour ces simulations. Ce cadre doit être validé par la réalisation d'un *atelier de génie logiciel pour simulations*.

2.2 Comportements

Comportement élémentaire. *Un* comportement élémentaire *est une séquence (donc ordonnée) d'actions qui, à partir d'une situation initiale donnée, permet à l'agent d'atteindre un objectif.*

Un comportement élémentaire est calculé par un moteur, appelé moteur de comportements ou moteur comportemental. Un comportement élémentaire est en fait un chemin de preuve calculé par le moteur dans le but de satisfaire l'objectif.

Comportement d'un agent *On appelle* comportement d'un agent, *l'ensemble des comportements élémentaires que l'agent (son moteur) est capable de produire/calculer pour tous les situations initiales et tous les objectifs possibles.*

Le comportement de l'agent est donc l'ensemble des chemins de preuves calculables par le moteur. Pour un moteur de comportements donné, un chemin de preuve calculé pour un objectif est dépendant du contexte, ie. l'environnement, et des propriétés de l'agent : propriétés factuelles, mais aussi capacités de l'agent. On parle donc de *moteur de comportements générique* à propos d'un moteur qui permet par simples variations de paramètres au niveau de l'agent d'obtenir des comportements différents. Par la suite nous n'utiliserons plus que le seul terme de comportement pour désigner ces deux notions. Il nous reste à préciser l'expression *comportement rationnel* (nous aurions également pu utiliser les qualificatifs *vraisemblable* ou *normal*). Ce que nous entendons par "rationnel", c'est que lors de l'exécution d'un comportement calculé, celui-ci ne paraisse pas à un observateur extérieur absurde ou irrationnel. Nous cherchons donc à ce qu'un spectateur de nos simulations ne conclue pas "le comportement de cet agent est anormal ou aburde" (sauf si c'était le but du comportement simulé évidemment). Ce critère de rationnalité est donc mesurer à l'aune du jugement d'un observateur (humain) qui vérifiera donc la conformité du comportement (apparant) de la simulation avec ce que lui pense comme étant un comportement acceptable pour la situtation décrite par la simulation.

2.3 Simulations

La notion de simulation sous-entend une tentative de reproduire plus ou moins précisément une réalité. Ceci implique qu'une modélisation de cette réalité est nécessaire. Lors de cette modélisation un certain nombre de simplifications sont le plus souvent effectuées (car indispensables) et du degré de ces simplifications dépendra le niveau de réalisme de la simulation. Nous parlons alors de *réalité adaptée*.

Les réalités qui nous intéressent sont celles faisant intervenir un certain nombre d'entités physiques évoluant dans un environnement muni d'une ou plusieurs métriques et obéissant à certaines règles de fonctionnement.

La modélisation opère à deux niveaux. D'une part il faut traduire les différentes entités physiques prenant part à la réalité considérée. D'autre part il faut exprimer les règles qui régissent l'univers modélisé et les actions qu'elles autorisent ainsi que leurs conséquences (réponses). Il s'agit donc bien évidemment ici de représenter la connaissance : factuelle pour la modélisation des entités, plus opérationnelle pour les règles. C'est cette connaissance qui est utilisée par un moteur comportemental pour effectuer ses calculs.

Simuler un comportement. *La* simulation d'un comportement *c'est exécuter chacune des actions qui le compose et donc calculer et appliquer les conséquences telles qu'elles ont été définies dans le cadre de la modélisation.*

À chaque étape la validité du plan (donc de ses actions) relativement aux règles de la réalité adaptée définie par la modélisation doit être vérifiée. Un comportement calculé peut donc être remis en cause une fois confrontée à la "réalité" de la simulation.

Simulation. *Une* simulation *est obtenue par la réunion des simulations des comportements de tous les agents impliqués.*

2.4 Généricité

L'objectif que nous affichons est de fournir un cadre *générique* pour la création de simulations, et donc pour les comportements. À travers ce terme *générique*, nous entendons la possibilité d'obtenir des simulations différentes construites à partir d'un seul et même formalisme. "Simulations différentes" signifient à la fois plusieurs simulations concernant des réalités différentes que plusieurs variations de simulations pour une même réalité adaptée. Nous proposons donc un formalisme permettant la modélisation d'un large éventail de réalités et des règles qui les régissent, ainsi qu'un moteur générique capable de prendre en compte le contexte d'évolution de l'agent pour fournir un comportement adapté et individualisé.

3 Notre modèle

Nous allons dans la suite de cette section décrire les différents éléments nécessaires à la modélisation de nos simulations. La plupart des idées devrait paraître comme "allant de soi" ou comme "étant de bon sens", mais, à notre connaissance, elles n'ont jamais été présentées ensemble pour fournir un cadre global et de plus l'une des difficultés consistent justement en l'agencement de ces différentes notions.

3.1 Environnement

L'environnement décrit la géographie de la simulation, il permet de situer les agents et ainsi de contrôler la possibilité de réalisation de certaines de leurs actions, lorsque la notion de proximité à une importance par exemple. L'environnement permet de donner un sens à la notion de *déplacement* d'un agent, bien qu'anodine cette notion est primordiale car elle influe énormément sur la dynamique, au moins apparente, de la simulation et nous distingue des problèmes de planification "simple" comme le "Singe-Bananes. C'est également grâce à l'environnement que les informations susceptibles d'être perçues par un agent peuvent être déterminées.

Nous représentons l'environnement par un graphe dont les nœuds sont des *places* et les arcs représentent les passages d'une place à une autre.

environnement = {place,passage*}*

Une place est une zone géographique élémentaire. La granularité d'une place est fonction de la simulation, la seule contrainte est qu'au sein d'une place il n'y a pas de restriction ni pour le déplacement, ni pour la perception (à l'exception des restrictions dues aux autres agents comme les problèmes de collisions par exemple). Une place peut représenter une pièce, une ville ou toute autre subdivision de l'environnement, et au sein d'une place la position d'un agent peut être gérée de manière discrétisée ou continue selon les besoins.

3.2 Agents

Nos agents sont localisés dans une place de l'environnement. C'est l'environnement qui est chargé de la création ou disparition d'un agent dans la simulation, même si la décision de ces créations ou disparition résulte a priori de comportements d'agents présents.

Nous appelons *agent* toute entité ayant une importance dans une simulation, c'est-à-dire dont la présence peut avoir une influence sur la simulation. Dans ces agents nous distinguons deux classes particulières qui sont les *agents passifs* et les *agents (pro-)actifs*. Nous utilisons également les termes d'*agents inanimés* et d'*agents animés*. C'est pour ces derniers que nous pourrons effectivement parler de comportement. D'une manière assez naturelle et habituelle, les agents seront représentés par des ensembles de propriétés, une propriété étant un couple (nom, valeur).

Nous allons cependant affiner cette définition beaucoup trop vague ou vaste en précisant certaines propriétés particulières imposées à nos agents. Nous passerons sur la propriété *nom* qui permet d'avoir une désignation symbolique de l'agent, pour nous concentrer sur ce qui permet de caractériser nos agents : les actions qui les concernent. Ainsi, nos agents (passifs ou actifs) sont en premier lieu caractérisés par les actions (nous utiliserons par la suite plutôt le terme "*interaction*" qui représente en fait la manière dont est codée une action) qu'ils peuvent subir. Nous nommons *peut-subir* cette propriété, la valeur attachée est la liste des interactions que peut subir l'agent (c'est-à-dire dont il peut être la cible), nous présenterons les interactions dans la section suivante.

Nous devons maintenant étudier le cas particulier des agents actifs. On le devine aisément ces

agents auront la possibilité d'agir sur leur environnement, c'est-à-dire sur les autres agents (vus sous leur facette "passif"). Ces capacités seront exprimées par une collection des interactions qu'ils peuvent effectuer, définie dans une propriété ; nous nommons *peut-effectuer* cette propriété. Cette propriété reste cependant une déclaration de compétences. Pour qu'un agent actif ait effectivement une influence dans la simulation, celui dispose d'un moteur de comportement qui a pour charge de décider à chaque instant de l'action que va accomplir l'agent, et donc de l'interaction qu'il va utiliser, en fonction du contexte. Ce moteur peut être influencé/dirigé par l'existence de buts/intentions propres à l'agent.

La version finale d'*agent* est donc celle de la figure 1 page 6. Sans modifier cette définition, nous pouvons mettre en évidence une propriété particulière qu'est la mémoire de l'agent actif. Celle-ci représente la base des connaissances accumulées par l'agent sur l'environnement : topologie de l'environnement, les autres agents (leur état et leur position). La non monotonie implique qu'il est nécessaire que ces informations soit datées. Ces informations correspondent en fait à la vision qu'a l'agent de l'environnement, il s'agit donc d'un environnement dégradé.

3.3 Les Interactions

Les interactions sont au centre de la modélisation de nos simulations. Nous pourrions d'ailleurs parler de *simulations orientées interactions* pour les définir. Ces interactions vont être à la base de la représentation des connaissances dans la simulation. Les autres propriétés des agents (si l'on exclut le moteur des agents animés) ne sont essentiellement que des connaissances factuelles. Les interactions caractérisent effectivement les actions qui peuvent être entreprises dans la simulation et en conséquence codent les lois de l'univers modélisé. Une interaction est paramétrée par un *acteur* et une *cible*. L'*acteur* pourra être instancié par un agent animé qui peut effectuer cette interaction et la *cible* est un agent qui peut subir cette interaction. Une interaction est définie de manière assez classique ainsi :

interaction = (nom, garde, condition, action)

Représentation de connaissances et généricité Ces interactions sont des unités de connaissance décrivant les lois du monde modélisé. Elles représentent une connaissance déclarative. Une conséquence est qu'une interaction ne doit pas, sauf cas très particulier, être attachée à une simulation mais doit représenter une connaissance relativement universelle. Cela représente une difficulté au niveau de la représentation de ces interactions, mais c'est ce qui permet la réutilisation d'une simulation à une autre. Le plus généralement une action a pour effet un changement d'état sur la cible ou l'acteur. Ainsi le fait d'*ouvrir* un objet (porte, coffre, fenêtre, etc.) a pour effet de le faire passer de l'état *ouvert* à l'état *fermé*. Peu importe ici la nature de la cible considérée.

Une telle interaction peut être utilisée par un moteur pour générer un plan tel que : "*pour franchir un obstacle ouvrable, il faut ouvrir cet obstacle*" (ou plus précisément la connaissance va être *il faut que cet obstacle soit ouvert*, ce qui se traduit quand ce n'est pas vérifié par le plan indiqué). Que cet obstacle soit une porte ou une fenêtre le plan est valide.

3.4 Dynamique de la simulation

La dynamique d'une simulation se fait grâce aux comportements des agents animés. En fait c'est la réunion de ces comportements qui produit la simulation. Notre objectif est d'autoriser le développement d'autant de comportements différents que possible. Mais pour atteindre cet objectif nous souhaitons qu'un seul et même (modèle de) moteur soit utilisé pour tous les agents actifs. Les différences entre les comportements ne doivent être dues qu'à des différences au niveau des propriétés des agents, et notamment les `peut-effectuer` et `buts`. Le moteur de comportement d'un agent (pour être précis il faudrait dire "l'instance du moteur de comportement pour un agent") est guidé par les buts de cet agent. Un but est décrit par une action à accomplir ou une condition à satisfaire. Un but peut être récurrent ou "à usage unique".

Le moteur de comportement doit effectuer la planification permettant la résolution des buts de l'agent. D'une manière assez naturelle, il s'appuiera sur une procédure de résolution du type "chaînage arrière" sur les interactions en s'appuyant sur les interactions qu'il peut effectuer. Ce chaînage doit aboutir à la détermination de la prochaine (inter)action à exécuter par l'agent en planifiant une séquence d'interactions devant permettre la satisfaction du but (on a donc un découpage en sous-buts, etc.). Plusieurs buts peuvent a priori se trouver en concurrence, un

agent = *agent-passif* | *agent-actif*
agent-passif= *{ ("peut-subir", {interaction* }), propriété* }*
agent-actif = *agent-passif* ∪ *{ ("peut-effectuer", {interaction*}), ("buts", but*), ("moteur", moteur)}*

FIG. 1: Définition finale de l'agent

critère de résolution de conflits doit donc permettre la sélection d'un but courant privilégié, considéré comme plus prioritaire à cet instant. Les problèmes qui se posent à ce niveau sont essentiellement dus au fait que le moteur doit s'appuyer sur une connaissance incomplète de l'environnement par l'agent. L'état ou la position des différents agents de la simulation ne peuvent être supposées connues. De plus l'évolution de l'environnement implique de considérer que l'environnement est non monotone. Un plan construit dans ces conditions peut donc se voir remis en cause et devra donc peut être être adapté pendant son exécution car certaines des étapes intermédiaires peuvent échouer ou, du moins, nécessiter un recalcul du plan. Une fois déterminée par son moteur l'interaction à exécuter par l'agent, il faut trouver une cible adéquate. Cette cible pouvant être plus ou moins précisée à ce moment : il peut s'agir d'un agent donné, ou d'une expression caractérisant un ensemble d'agents acceptables (le premier cas est un cas particulier de celui-ci). Par exemple pour exécuter une interaction `manger`, un agent pourrait chercher toute cible "mangeable", il s'agit alors de l'ensemble $\{a \in Agent, \texttt{"manger"} \in a.peut - subir()\}$. On obtient alors le comportement suivant :

Soit $a \in$ Animé tq a veut effectuer l'interaction i (ce qui nécessite $i \in$ a.peut-effectuer()) le comportement de a est de ***chercher*** *dans l'environnement un agent $t \in$ Agent tq $i \in$ t.peut-subir() et ensuite de tenter de résoudre $i(a, t)$.*

Une difficulté majeure que nous n'avons pas la place d'aborder ici est la nécessité dans de telles simulations concernant des agents situés, de prendre en compte les déplacements des agents.

4 Conclusion

La réalisation de simulations de comportements qui seront perçues et jugées comme rationnelles par un observateur humain n'est pas un problème simple. Les solutions apportées par les systèmes réactifs ne sont pas satisfaisantes d'un point de vue généricité et donc réutilisabilité. Guidé par ces soucis de Génie Logiciel, nous nous somme attachés à proposer un modèle général pour la simulation de comportements. La dynamique de ce modèle s'appuie sur la description d'interactions que peuvent subir et peuvent effectuer des agents situés au sein d'un environnement dynamique. Un moteur comportemental générique utilise ces interactions pour proposer un plan d'actions aux agents, en tenant compte des contraintes liées à la non monotonie de l'environnement et à la prise en compte du caractère situé des agents.

Références

[1] Rodney A. Brooks, Cynthia Breazeal, Matthew Marjanović, Brian Scassellati, and Matthew M. Williamson. The COG Project : Building a Humanoid Robot. In C. Nehaniv, editor, *Computation for Metaphors, Analogy, and Agents*, volume 1562, pages 52–87. Springer Verlag, 1999.

[2] Kerstin Dautenhahn. The role of interactive conceptions of intelligence and life in cognitive technology. In J. P. Marsh, C. L. Nehaniv, and B. Gorayska, editors, *Proceedings of the 2nd International Cognitive Technology Conference (ICTC'97)*. IEEE Press, 1997.

[3] Daniel C. Dennett. *La stratégie de l'interprète : le sens commun et l'univers quotidien. nrf* Essais. Gallimard, Paris, 1990. Traduit de l'anglais par Pascal Engel.

[4] Jean-Louis Dessalles. *Aux origines du langage : une histoire naturelle de la parole*. Hermès, Paris, 2000.

[5] J. Erceau and Jacques Ferber. L'Intelligence Artificielle Distribuée. *La Recherche*, 233, 1991.

[6] John E. Laird and Michael van Lent. Human-level AI's Killer Application : Interactive Computer Games. AAAI Press, 2000.

[7] Léna Soler. *Introduction à l'épistémologie*. Ellipses, 2000.