Un applicatif de modélisation individus-centrée fondé sur une architecture de type multi-agents (MUFINS)

Jean-Christophe Soulié, David Guyomard, Martin Desruisseaux, Jean-Michel Stretta, Michel Petit

© D. Guyomard

À la suite des démarches statistiques exploratoires, qui ont permis de dégager notamment les principales relations pouvant exister entre les paramètres de l'environnement océanique et les résultats des pêches, une approche de modélisation des comportements des grands pélagiques dans le paysage océanique tropical du sud-ouest de l'océan Indien est proposée.

Deux domaines nous ont semblé particulièrement porteurs de concepts et de méthodes originales : l'écologie du paysage et la modélisation multi-agents. Nous en présentons ici les aspects les plus pertinents au regard de notre problématique.

Le modèle MUFINS (*MultiFish Indian ocean Simulator*)

Les avancées permises par la modélisation individus-centrée et les SMA sont d'évidence à leurs prémices. Les techniques de ce type de modèle sont exposées plus haut. Il serait particulièrement intéressant d'intégrer plus systématiquement des données d'environnement réelles dans la description de l'environnement de simulation et de pouvoir considérer cet environnement dans sa composante

structurelle fine et sa dynamique. L'intégration de multiples couches d'informations (le modèle de DAGORN, 1994, ne proposait que la température de surface de la mer comme variable descriptive du milieu) au sein d'une architecture de simulation individus-centrée est aussi souhaitable. Il s'agirait alors de simuler des dynamiques comportementales individus-centrées au sein d'un système d'information géographique (SIG) dynamique.

Capturabilité palangrière

L'étude des déplacements individuels de l'espadon dans le contexte environnemental fourni par les cartes satellitaires est le premier objectif de ce travail. Il se situe ainsi dans le domaine de l'« éthologie synthétique » (DROGOUL, 2000), afin d'apporter des éléments de compréhension de la dynamique de la population de poissons étudiée. Il propose une contribution à la compréhension des mécanismes intervenant dans la réalisation de structures collectives à méso-échelle (au sens de la distribution spatiale de la ressource) à partir du traitement de l'information intervenant à l'échelle du poisson individuel. Les effets qui déterminent le transfert jusqu'à l'échelle océanique (celle de la population globale d'espadons) ne seront que peu abordés ici.

L'objectif d'une recherche halieutique est avant tout de proposer des éléments d'analyse et de suivi de l'activité de pêche, dans une optique de gestion durable de la ressource. À l'issue de la 1re partie, il apparaît qu'à méso-échelle, la problématique de la capturabilité différentielle de l'espadon par la palangre de surface constitue un point important de la définition d'indices d'abondance pertinents.

C'est pourquoi nous proposons d'introduire explicitement le processus de capture dans le modèle d'écologie synthétique. Aux agents « animats » qui constitueront la population d'espadons simulés au sein du paysage environnemental, s'adjoindront donc des agents « lignes de pêche », situés et dotés d'un comportement propre. Les positions de ces lignes de pêche seront les positions réelles des lignes de pêche de la pêcherie palangrière réunionnaise des années 1998 à 2000, qui seront insérées dans le modèle en cours de simulation, aux dates correspondantes.

Ces agents « lignes de pêche » sont alors dotés de la capacité de percevoir les agents « animats espadon » dans l'espace topologique du modèle, puis de mettre en œuvre un comportement de « capture virtuelle » de ces animats. Il s'agit ainsi d'explicitement modéliser le processus d'accessibilité du poisson par la palangre, la composante liée aux interactions avec l'engin de pêche (vulnérabilité) étant considérée comme constante dans le modèle (du fait de l'homogénéité des pratiques de pêche). Les aspects liés à la correspondance entre l'habitat vertical de l'espadon seront pris en compte, et seront intégrés à cette composante d'accessibilité.

Ainsi, les captures virtuelles constitueront une forme de validation des déplacements individuels des animats, en les comparant avec les captures réelles de la pêcherie. L'utilisation des captures comme validation des comportements

des animats est toutefois soumise à plusieurs contraintes : la composante de vulnérabilité n'est pas considérée (les effets d'interaction avec l'engin de pêche constituent une part substantielle de la capturabilité ; Bach, *comm. pers.*) et l'effort de pêche n'étant pas distribué de manière aléatoire et homogène sur toute la zone exploitée, il ne permettra pas de mettre en évidence les phénomènes sur toute cette zone. En l'absence de données directes de déplacement des espadons dans la zone (marquage, acoustique et marque-archive), les captures restent les seuls indicateurs indirects de la présence d'espadons.

Caractéristiques générales du modèle

Le modèle proposé repose donc sur quatre postulats fondamentaux :

– l'information environnementale est perçue de manière directe par les poissons ;

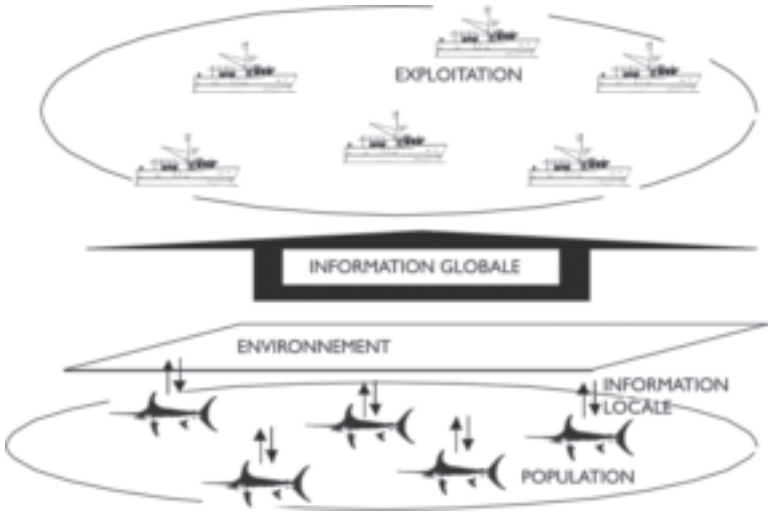
– c'est cette information structurelle qui détermine leurs déplacements dans le milieu, à méso-échelle, sous contraintes trophiques et reproductives ;

– la fonctionnalité de l'environnement (telle que perçue par les pêcheurs) « émerge » de l'ensemble de ces comportements individuels ;

– la pêche apporte les éléments de validation et la connaissance globale de l'environnement permet de caractériser le paysage environnemental d'un point de vue fonctionnel (stratification spatiale de l'environnement).

On peut résumer cette approche par le modèle d'information présenté à la figure 100.

Modéliser un poisson « *de l'intérieur* » pourrait apporter des réponses originales aux questions posées par la distribution des ressources dans leur environnement.



▽ *Fig. 100*
Le modèle d'information.

Pour ce faire, nous disposons de connaissances que les biologistes et éthologues ont pu accumuler sur les espèces, susceptibles de fournir des hypothèses pertinentes sur les processus étudiés. C'est ainsi que certains auteurs évoquent l'étude de « l'énaction[1] d'un monde propre » (*Umwelt*) de l'animal (MATURANA et VARELA, 1994 ; THERAULAZ et SPITZ, 1994). Ce monde propre représente l'ensemble des significations que l'individu animal peut attribuer aux formes rencontrées, et auxquelles il peut répondre.

Nous proposons donc un outil de modélisation individus-centrée développé comme une plate-forme de simulation de type multi-agents pour la simulation du comportement de grands pélagiques (ici appliquée à l'espadon) et de l'exploitation halieutique (ici appliquée à la pêche palangrière réunionnaise) au sein d'un environnement multi-paramètres (ici appliquée aux paramètres de l'environnement océanique de surface captés par satellite). Notre approche se situe donc résolument comme une synthèse des éléments de la modélisation individus-centrée (des animats espadons se comportent dans un espace virtuel selon des hypothèses éthologiques et écologiques), de l'écologie du paysage (l'environnement est représenté par différents paramètres dont on cherche à comprendre les relations horizontales à travers le comportement du poisson), des systèmes multi-agents (des agents « espadons » et des agents « lignes de pêche ») et de la vie artificielle (modélisation de phénomènes « vivants » et émergence d'objets océaniques fonctionnellement cohérents à partir d'hypothèses sur le comportement d'agents poissons).

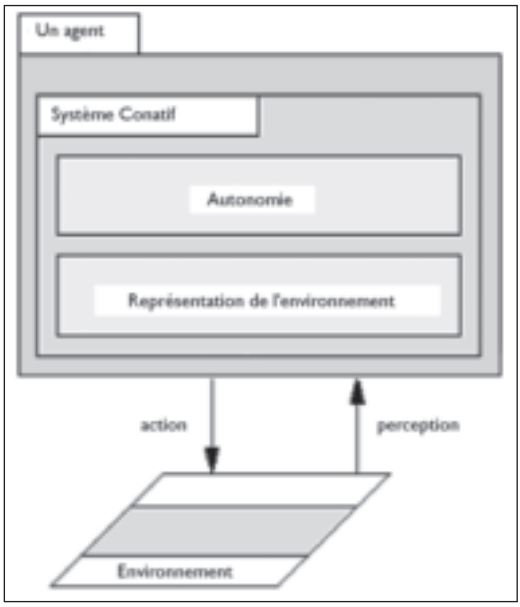
À partir des objectifs et des contraintes de simulation exposées en 2e partie, un modèle informatique a été conçu et implémenté. La version opérationnelle de ce modèle, MUFINS, s'applique à la simulation de déplacements d'espadons virtuels au sein d'un environnement multi-paramètres décrit par les données satellitaires du sud-ouest de l'océan Indien.

En effet, les aspects de gestion de l'environnement, particulièrement lorsque celui-ci peut-être caractérisé par plusieurs « couches » d'information (à la manière d'un SIG), avaient auparavant été peu abordés dans le domaine de la modélisation multi-agents. La composante dynamique de l'environnement introduit une difficulté supplémentaire à la conception du modèle, qui doit explicitement gérer le déroulement du temps, le renouvellement des cartes et des états des agents, à la manière d'un SIG dynamique (SOULIÉ, 2001 ; SOULIÉ *et al.*, 2001).

Du modèle d'agent…

FERBER (1997) a défini un modèle général pour représenter et gérer les interactions ainsi que les priorités entre les agents et l'environnement (fig. 101). SOULIÉ (2001) s'est attaché à construire un modèle qui, à partir de l'approche proposée par Ferber, permet de gérer des environnements multiples pour un ou plusieurs agents simultanément.

1. Dans le paradigme proposé par l'énaction, l'agent cognitif et le monde qu'il « vit » forment un ensemble cohérent où l'un et l'autre se définissent mutuellement (ROUET, 1998).



▽ *Fig. 101*
Le modèle d'agent de FERBER *(1997).*

SOULIÉ (2001) souligne la distinction, classiquement établie par les analystes et programmeurs informatiques, entre systèmes multi-agents purement communiquants et situés.

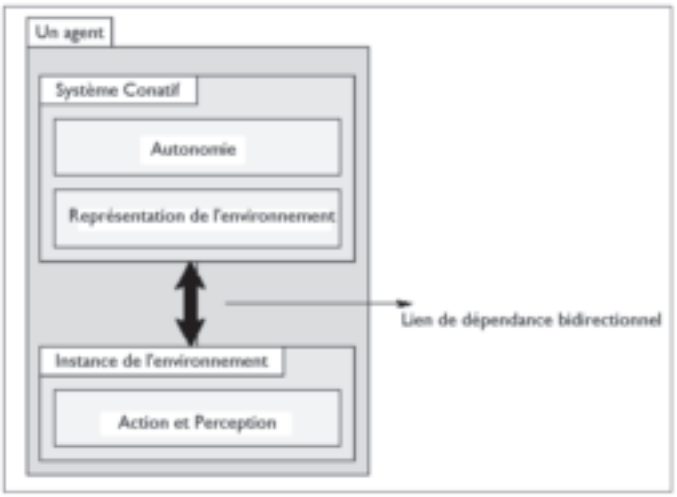
- Lorsque l'environnement est vide et que celui-ci contient des objets, on a un système multi-agents *purement communiquant.*
- Lorsque l'environnement est muni d'une métrique ainsi que d'un espace métrique, on a un système multi-agents *situé*.

L'objectif du modèle est justement de permettre de gérer des agents « animats » et des agents « lignes de pêche » qui appartiennent à la fois à un système multi-agents communiquant et à la fois à un système multi-agents situé. La présence simultanée d'un réseau d'accointances (entre agents) et d'un environnement réel (ici « multicouches ») nécessite de constituer une architecture qui ait des caractéristiques des deux systèmes.

Pour ce faire, SOULIÉ (2001) a, à partir du schéma du modèle général de FERBER (1997), construit ce nouveau modèle, dit *multi-environnemental*, en plusieurs étapes : séparation des environnements, accès aux données, gestion du temps, maintien de l'intégrité des données et modifications du système conatif lié à l'apport des environnement multiples. Nous tâcherons de mettre en évidence le déroulement de ces étapes au cours des paragraphes suivants qui présentent le modèle et son fonctionnement. Les diagrammes complets des classes de MUFINS et l'organisation des packages sont disponibles en annexe.

… au modèle multi-environnemental

Soulié propose ainsi une définition « élargie » de l'agent de Ferber (fig. 102). L'agent est ici doté d'un *système conatif* et d'une *instance dans l'environnement* chargée de l'*action* et de la *perception*. Le système conatif est chargé de la partie raisonnement et de l'autonomie, alors que l'instance dans l'environnement gère tout ce qui a trait à l'environnement uniquement. Entre ces deux entités, le *lien bidirectionnel de dépendance* permet de faire transiter des informations.

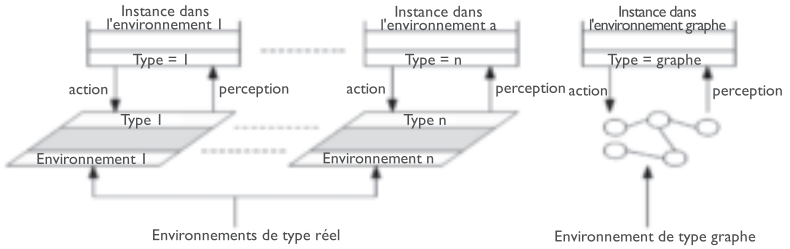


▽ *Fig. 102*
Un agent avec ses deux parties et son lien de dépendance.

Pour pouvoir agir sur plusieurs couches d'informations environnementales, considérées chacune comme un environnement réel *typé*, il est nécessaire de créer autant d'instances dans l'environnement que de types d'environnements que l'on doit gérer. Le type de l'environnement est constitué d'une simple chaîne de caractères.

Les types d'environnement réels ici correspondent aux différents paramètres océanographiques décrits précédemment, mais aussi aux modifications appliquées à ces paramètres. L'environnement constitué par les positions des animats eux-mêmes, tels que perçus par les agents lignes de pêche, peut être assimilé à un environnement réel typé mais son fonctionnement est bien sûr différent.

Le modèle multi-environnemental générique consiste donc dans un premier temps à créer pour chaque environnement de type t_i une instance dans l'environnement qui va correspondre à ce type t_i. Les principes d'actions et de perceptions sont conservés car tous les capteurs et les effecteurs qui sont contenus dans une instance dans l'environnement de type t_i sont effectivement capables d'agir sur l'environnement t_i. La figure 103 illustre cette première décomposition.



▽ *Fig. 103*
Le découpage des environnements dans MUFINS.

Afin de pouvoir faire remonter les informations perçues par les instances de l'environnement grâce à leurs capteurs, et, inversement, que le système conatif puisse envoyer des commandes aux instances de l'environnement, il faut créer pour chaque agent autant de liens de dépendances bidirectionnels que d'environnements. Chaque lien bidirectionnel sera ainsi typé, du même type que l'environnement avec lequel il communique. Il pourrait y avoir au moins un lien bidirectionnel pour un environnement réel et un lien bidirectionnel pour un environnement de type graphe, par exemple.

Le diagramme des classes génériques (package `mufins/mas/agent`) illustre l'implémentation du simulateur, avec autant de classes que d'éléments spécialisés du modèle d'agent. Un agent créé devra ainsi instancier autant d'éléments que de classes illustrées, et autant que de types d'environnements. Un seul environnement n'est pas typé, il sera présenté ultérieurement dans ce chapitre.

Cette décomposition résulte d'une analogie extrêmement intuitive avec le fonctionnement du système nerveux sensoriel et moteur des vertébrés supérieurs : des cellules spécialisées génèrent, transportent et traitent l'influx nerveux entre les capteurs dédiés dans l'environnement et le cerveau qui traite l'information, qui commande en retour une action sur l'environnement. On peut considérer que chaque classe du modèle correspond à une de ces cellules spécialisées.

De ce diagramme générique, deux packages ont hérité, correspondant aux agents « animats espadons » et agents « lignes de pêche ». Les agents sont spécialisés au sein des classes filles héritées. Ainsi, une instance dans l'environnement d'un animat sera caractérisée par une variable de position point (deux coordonnées, latitude et longitude) tandis qu'une instance d'une ligne de pêche sera positionnée par deux points (positions de début et de fin de filage).

La société des agents rassemble l'ensemble des agents animats et lignes de pêche, et s'occupe de tenir le registre des entrées/sorties des agents du système.

L'accès aux données

Chaque type d'environnement nécessite un accès aux données propres qui le caractérisent. Ici, les données auxquelles ont besoin d'accéder les animats sont

d'abord les images satellitaires, mais les lignes de pêche doivent aussi accéder aux positions des animats pour simuler le processus d'accessibilité. L'architecture générale de MUFINS permet de gérer des données de type graphe, des données provenant d'autres logiciels ou tout simplement des données codées directement dans le simulateur ou calculées et stockées en cours de simulation.

Chaque environnement étant typé, l'accès aux données liées à cet environnement va se faire pour le type de données de cet environnement. Chaque environnement du modèle sera doté d'une *interface d'accès aux données* qui va faire le lien entre les données qui vont être nécessaires pour les agents et les environnements.

Cette interface devra gérer les données stockées dans des bases de données ou des fichiers (et disposer des primitives permettant d'y accéder), et les données calculées en temps réel à chaque pas de simulation. Dans le cas d'un environnement de type réseau d'accointances, l'interface d'accès aux données sera capable de fournir à l'environnement les données liées à l'évolution du graphe à chaque fois que ce sera nécessaire. Ce cas n'a pas été explicitement implémenté dans notre application, mais l'architecture générale le permet.

Dans MUFINS, les données auxquelles accèdent les agents sont donc :

– les données satellitaires typées : SST, SLA, Ekman, Chl-*a*, Bathy, Courants de surface, et produits dérivés (gradients) ;

– les données de position des agents : positions ponctuelles des animats et positions des lignes de pêche.

Accès aux données satellitaires

Une classe dédiée à la connexion avec la base de données Seas s'occupe de récupérer l'image satellitaire correspondant à la date courante de déroulement de la simulation (classe `SeasConnectivity`, héritée de `DataBaseConnectivity`). C'est cette classe qui ouvre la connexion JDBC avec la base Seas et exécute les requêtes appropriées. Le type de l'image peut être soit le nom du paramètre simplifié (SST, SLA, Chl-*a*, Ekman, CGU, CGV, Bathy), soit un nom de paramètre modifié par les fonctions présentées en 2e partie. Dans ce cas, l'image renvoyée est différente : soit un filtre convolution lui est appliqué (pour obtenir les valeurs de gradients selon l'opérateur de Sobel), soit l'image renvoyée correspond à la date antérieure ou postérieure précisée dans le type.

Différentes méthodes donnent accès aux valeurs des paramètres au sein des images. Chaque animat récupère systématiquement la valeur du paramètre de chaque environnement à sa position géographique courante dans le pixel correspondant à cette position. Mais le système conatif de l'agent dispose de méthodes plus élaborées (et plus performantes en termes de vitesse d'exécution) qui permettent de récupérer des « tuiles » de valeurs (et les positions des pixels correspondantes) : cela évite de multiplier les boucles de programmation (coûteuses en ressources système) pour évaluer le contexte d'un agent.

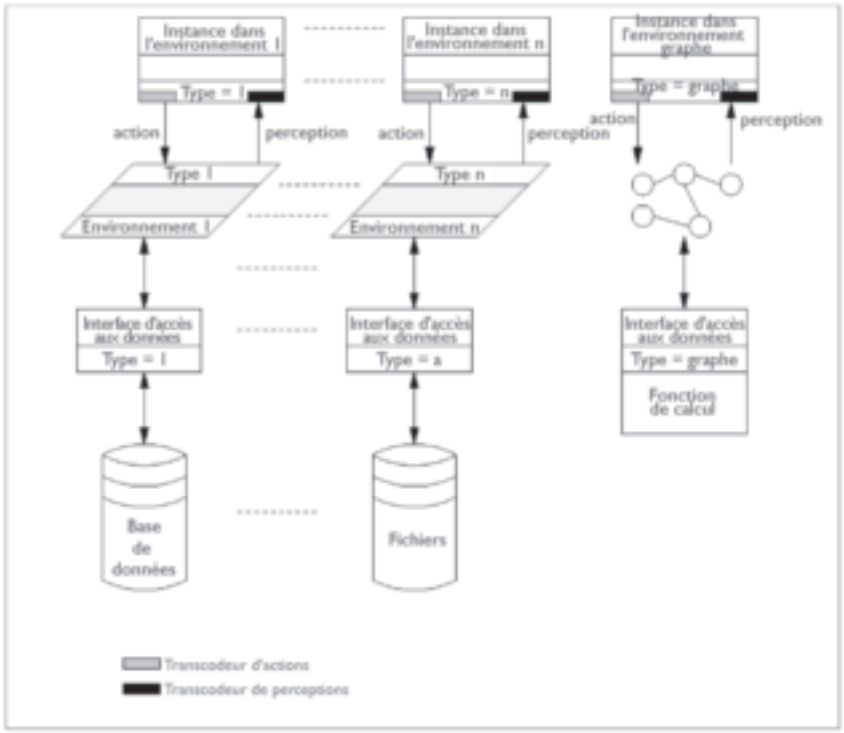
ACCÈS AUX DONNÉES DE POSITIONS DES AGENTS

Les positions des lignes de pêche sont stockées dans une base de données Fishery et accédées de la même manière que décrite précédemment (JDBC MySql) par la classe `FisheryConnectivity`.

Les agents lignes de pêche accèdent aussi aux positions des animats par une classe appelée `ResultsConnectivity`, qui sert aussi à insérer les positions successives des animats (et les valeurs des paramètres captées) au fur et à mesure de la simulation.

C'est une architecture semblable à une architecture par tableau noir (FERBER, 1997), la base de données contenant l'ensemble des données des états des autres agents. Toute forme d'interactions entre animats sera codée de la même manière.

La figure 104 présente la deuxième étape de la construction du modèle multi-environnements.



▽ *Fig. 104*
L'accès aux données dans MUFINS.

Le système conatif

Le système conatif d'un agent récupère donc facilement toute information de l'environnement à partir de ses instances dans les environnements réels. Il

traite ensuite cette information selon les règles de comportements qui lui sont spécifiées par l'utilisateur, présentées en détails dans le chapitre suivant.

Animats et comportements

Le système conatif reçoit des informations typées, encapsulant la valeur au pixel où se situe l'agent avec ses coordonnées géographiques en longitude/latitude. Certains comportements peuvent s'en accommoder comme source d'information avant de mettre en œuvre un comportement et d'envoyer une commande. Mais surtout, le transfert de cette information « marque » le début du processus décisionnel du système conatif dans le système de gestion du temps.

Le système conatif reçoit donc une information typée avec une position, et sa délibération en renvoie une avec une position de destination. C'est le rôle du système conatif que de « décider » d'une destination à chaque étape de simulation. Cette information est ensuite renvoyée aux instances des environnements réels (par l'intermédiaire de l'environnement virtuel, cf. *infra*), qui effectuent un déplacement au sein de chaque environnement réel.

Les comportements sont codés dans une classe *finale* qui va récupérer diverses données par l'intermédiaire de l'architecture multi-environnementale et du système d'accès aux données. Ainsi, chaque comportement établit le contexte environnemental dont il a besoin pour délibérer. Cela permet d'optimiser la quantité d'information qui doit transiter le long de l'architecture des classes selon le degré de complexité du comportement mis en œuvre.

Ainsi, certains comportements récupèrent l'ensemble des valeurs dans une zone située autour du point de départ, d'autres des valeurs captées par l'agent précédemment (mémoire de l'agent).

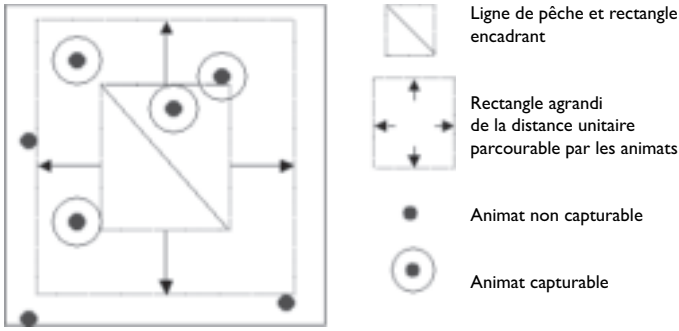
Lignes de pêche et capture virtuelle

Le comportement des lignes de pêche est différent du comportement des animats : les lignes ne se déplacent pas en cours de simulation, mais elles interagissent avec les animats par le processus de *capture virtuelle*.

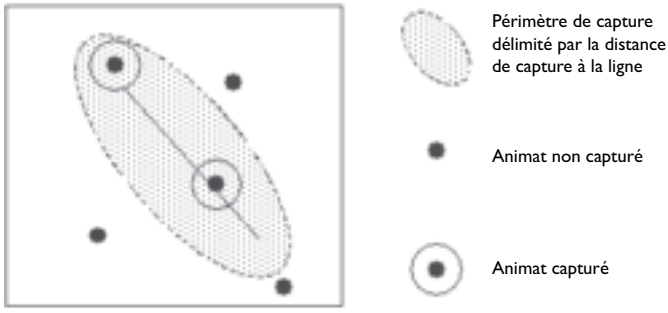
À tout instant de la simulation, les lignes de pêche peuvent percevoir les positions des animats au moyen de l'interface dédiée (connexion à la table *Positions* de la base *Simulations*). Le processus de capture virtuelle est basé sur les distances géographiques entre les lignes et les animats : si un animat est proche d'une ligne de pêche, il est considéré comme « capturable ».

Toutefois, pour optimiser les performances du simulateur, plusieurs options méthodologiques ont été choisies. Ainsi, chaque ligne de pêche ne récupère pas la totalité des distances de tous les animats présents dans la société des agents : un périmètre de capture est défini, en considérant le rectangle délimité par la ligne de pêche, élargi d'une distance correspondant grossièrement au déplacement possible des animats au cours de la durée de simulation pendant laquelle la capture est mise en œuvre. Ainsi, tous les animats présents dans ce périmètre élargi sont susceptibles d'être capturés par cette ligne (fig. 105).

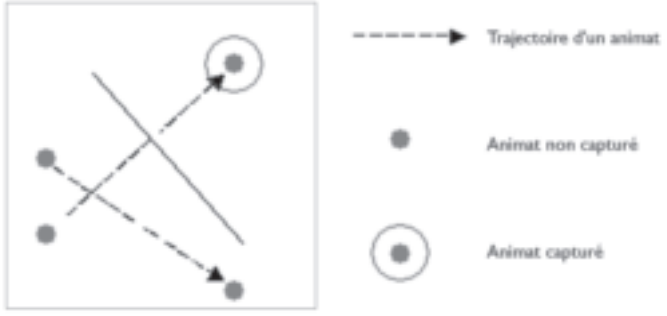
Une fois la liste des animats « capturables » disponible pour chaque ligne, deux types de processus de capture ont été codés.



▽ *Fig. 105*
Schéma de la sélection des animats « capturables » par les lignes de pêche.



▽ *Fig. 106*
Schéma du processus de capture statique.



▽ *Fig. 107*
Schéma du processus de capture dynamique.

Le premier type correspond à la capture caractérisée de *statique* (fig. 106). Dans ce cas, la distance de l'animat à la ligne seule conditionne la capture au pas de temps considéré. La distance de capture est spécifiée par l'utilisateur, en milles nautiques. Si un animat se trouve à une distance euclidienne inférieure à cette distance, il est alors capturé.

Le deuxième type de capture est appelé capture *dynamique* (fig. 107*)*. Cette fois, ce n'est pas la distance de l'animat à la ligne à un instant donné qui détermine la capture. Il est ici nécessaire de disposer de la trajectoire de chaque agent entre l'instant précédent et l'instant considéré : si cette trajectoire croise la ligne de pêche (intersection non nulle), l'animat est capturé.

Dans ce cas, les animats correspondants peuvent être enlevés du système selon la spécification de l'utilisateur.

Le maintien de l'intégrité des données et la gestion des conflits

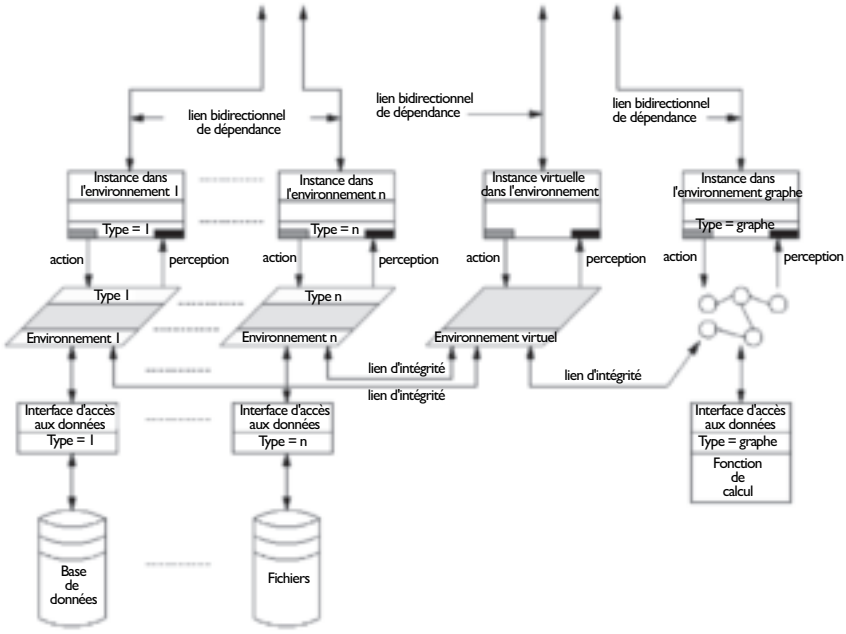
Le modèle d'agent (fig. 101, 102) montre comment les informations perçues dans l'environnement par les capteurs des instances environnementales remontent au système conatif de l'agent, qui « délibère » et renvoie alors les commandes aux effecteurs de ses mêmes instances environnementales par le lien bidirectionnel.

La commande passée envoie une position en coordonnées géographiques longitude/latitude. Mais elle n'est pas directement appliquée aux environnements réels typés. Dans le cas du déplacement d'un animat par exemple, il s'agit d'une action qui agit sur tous les environnements réels de la même manière. Envoyer la même commande à tous les environnements serait une perte de performances. On introduit alors un environnement particulier, déjà évoqué précédemment, l'*environnement virtuel*.

L'environnement virtuel a une connaissance de ce que sont et ce que représentent les autres environnements. Lors de l'initialisation du système, chaque environnement présent dans le système va s'identifier auprès de l'environnement virtuel grâce à un message spécial qui va transiter le long des *liens d'intégrité*.

Toute commande commune à tous les environnements, comme le déplacement d'un animat, est ainsi envoyée à l'environnement virtuel. Celui-ci va ensuite sélectionner les liens d'intégrités des environnements qui sont impliqués par ce changement de position (les environnements réels) et envoyer un message qui va prévenir les environnements que la position de l'agent à été modifiée. Ainsi, chaque agent dispose d'une instance dans l'environnement virtuel qui contient ses coordonnées géographiques longitude/latitude, alors que les instances dans les environnements réels sont caractérisées par des coordonnées en pixels, selon la résolution des données de chaque type d'environnement.

À cet effet, des *transformations affines* sont appliquées par l'environnement virtuel pour garantir l'intégrité des positions des agents dans tous les environnements réels, en fonction de leur propre système de coordonnées. De la même manière, toutes les coordonnées et distances géographiques (en milles) qui sont spécifiées par l'utilisateur (comme la distance de capture, par exemple) sont converties en distance en pixels dans les environnements réels concernés.



▽ *Fig. 108*
L'environnement virtuel et les liens d'intégrité dans MUFINS.

L'environnement virtuel est une classe mère des environnements instanciés typés, en plus d'être une classe instanciée qui sert à toutes les actions communes. Ce « double rôle » peut susciter une ambiguïté de compréhension, mais est en fait tout à fait logique dans le modèle objet de MUFINS. Cela permet notamment de limiter le trafic le long des liens de dépendance bidirectionnels dans le sens descendant (commandes) des actions communes (type déplacement).

Plus généralement, l'environnement virtuel sert ainsi à assurer l'intégrité des données. En cas de modification sur un environnement réel par l'action d'un agent, susceptible de modifier d'autres environnements, c'est l'environnement virtuel qui s'occupe de maintenir l'intégrité du système, en transmettant les messages *ad hoc* par les liens d'intégrité auprès des environnements réels concernés. Ici encore, l'implémentation de MUFINS n'a pas eu à considérer ce cas, mais il pourrait être envisagé dans des applications ultérieures. La gestion du temps est aussi assurée par l'environnement virtuel (cf. *infra*).

La figure 108 présente la troisième étape de la construction du modèle multi-environnements.

La gestion du temps

La gestion du temps dans MUFINS nécessite une bonne organisation des événements, car les environnements réels sont dynamiques, c'est-à-dire qu'ils doivent renouveler régulièrement leurs données en cours de simulation. De plus, il faut gérer les événements propres aux agents : déplacements des animats et captures virtuelles des lignes de pêche.

SOULIÉ (2001) a proposé un mécanisme de gestion du temps hybride entre la gestion dite *centralisée* du temps, et la gestion dite *dirigée par les événements*. Au sein de l'environnement virtuel, un objet *minuteur* s'occupe de garantir le déroulement des événements dans un ordre cohérent. Le temps est ainsi décomposé en *pas de temps* (en anglais, *step*), correspondant à des fractions égales de la journée de 24 heures : le nombre de pas de temps dans une journée dépend de la spécification de l'utilisateur.

Le minuteur centralise le déroulement des événements, en tenant à jour l'indice des pas de temps (à l'initialisation du système, il est égal à 0) et la date courante du système. Dans le cas des événements liés aux déplacements des animats, lorsque l'environnement virtuel ne reçoit plus de commandes de la part des animats et qu'il a distribué les commandes aux environnements réels, il prévient le minuteur de passer au pas de temps suivant.

L'environnement virtuel tient aussi les autres environnements informé de ce déroulement. Lorsque le nombre de pas de temps correspond à une journée, le minuteur avance la date d'un jour, et un message est passé aux environnements pour leur dire de se mettre à jour. Ce message est passé à tous les environnements, même si ceux-ci ne nécessitent pas de renouveler leurs données tous les jours (comme la SLA, la Chl-*a* ou la bathymétrie par exemple). C'est l'interface d'accès aux données par l'intermédiaire de la base de données des images (*Seas*) qui gère ensuite le renouvellement effectif des données : selon le type de données, la base sélectionne l'image qui correspond le mieux à la nouvelle date.

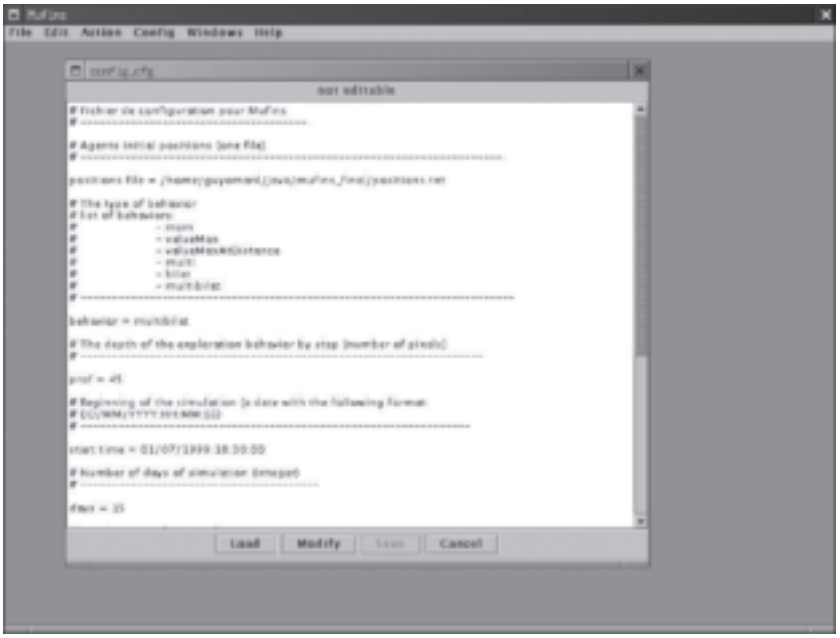
De la même manière, les agents lignes de pêche sont tenus au courant par le minuteur des pas de temps pendant lesquels ils doivent mettre en œuvre le processus de capture. En effet, il apparaît clairement que les captures d'espadon ne se répartissent pas également au cours de toute la journée, mais s'effectuent en majorité dans les premières heures de l'opération de pêche, correspondant aux dernières heures de la journée. Il est ainsi possible de modifier la proportion de pas de temps au cours d'une journée pendant lesquels les lignes de pêche sont « pêchantes » dans le système. Par exemple, pour un nombre de pas de temps journaliers de 24 (soit un pas de temps d'une durée d'une heure), on peut considérer que seules les dernières 6 heures de la journées sont « pêchantes », soit le quart de la journée. Ce paramètre peut être modifié aisément par l'utilisateur.

L'interface graphique de MUFINS

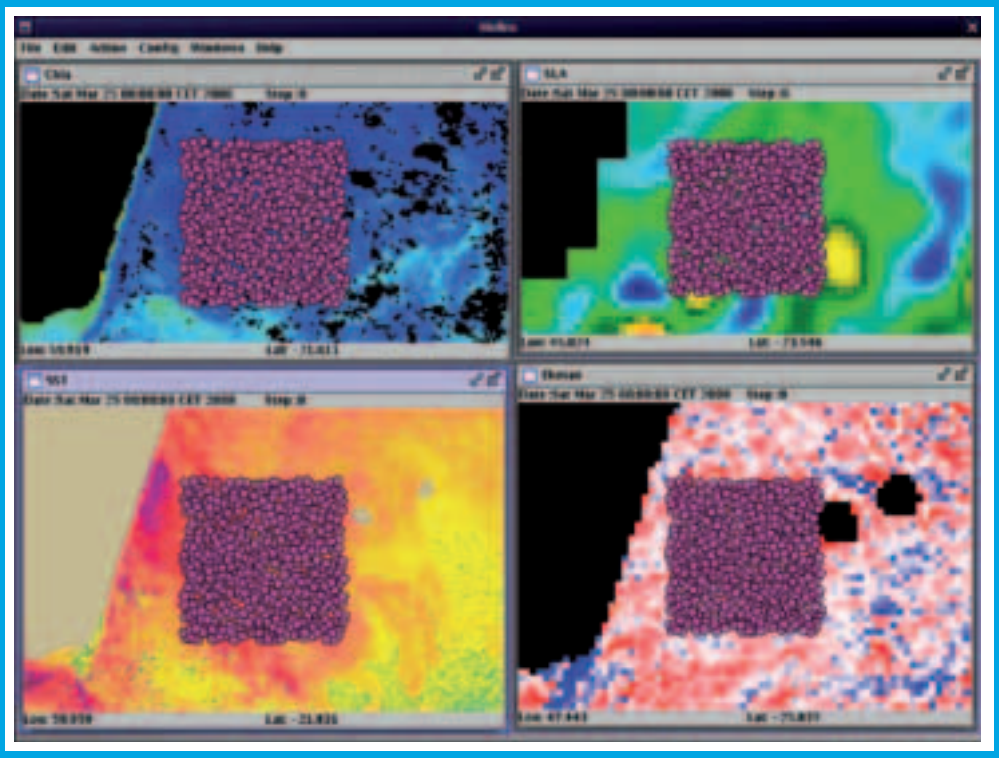
Une interface graphique a été implémentée pour permettre de suivre les simulations des comportements des agents sur les cartes satellitaires. Le package *mufins/gui* se charge de « suivre » les simulations pour mettre à jour les affichages (interface *Observer* et classe *Observable*).

La figure 109 montre ainsi l'interface graphique au chargement du fichier de configuration de MUFINS, qui spécifie les caractéristiques des agents (positions initiales, type de comportement, rayon d'exploration...) et des environnements (paramètres, dates de départ, nombre de jours de simulations, accès aux bases

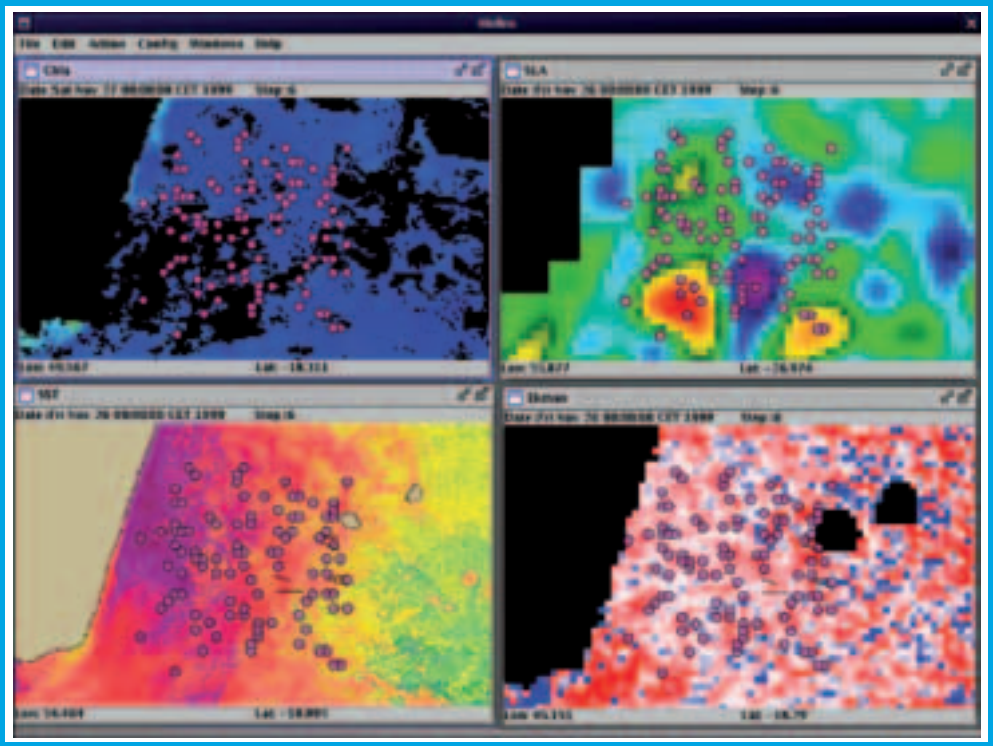
de données). La figure 110 (cf. cahier hors-texte) montre l'interface de simulation à l'initialisation des environnements (distribution régulière de 200 animats), avec quatre environnements ici disponibles et représentés (Chl-*a*, SLA, SST, Ekman). La figure 111 (cf. cahier hors-texte) montre enfin une vue d'écran de MUFINS en cours de simulation : les agents se distribuent de manière cohérente sur les différents environnements, et les lignes de pêche apparaissent explicitement.



▽ *Fig. 109*
Interface graphique de MUFINS : le fichier de configuration.



▲ *Fig. 110 - Interface graphique de MUFINS : à l'initialisation, en attente du départ de la simulation.*



▲ *Fig. 111 - Interface graphique de MUFINS : en cours de simulation (les traits fins représentent les lignes de pêche).*