Simulateur Prairie-Troupeau : Développement d'un module de défoliation contrôlée

Benoît Péan, Raphaël Martin

Institut Supérieur d'Informatique
de modélisation et de leurs Applications

COMPLEXE DES CEZEAUX
BP 125 – 63173 AUBIERE CEDEX

INRA-Unité d'Agronomie

Equipe Fonctionnement et
Gestion de l'Ecosystème Prairial

DOMAINE DE CROUEL
234, Avenue du Brézet
63039 CLERMONT-FERRAND CEDEX

Rapport de projet 2ème année 2003-2004

Simulateur Prairie-Troupeau : Développement d'un module de défoliation contrôlée

Présenté par : Raphaël MARTIN
Benoît PEAN

Durée : du 16/10/03 au 20/03/04

Responsable ISIMA : Mme Christine FORCE
Responsables INRA : M Pascal CARRERE
Mlle Frédérique LOUAULT

Institut Supérieur d'Informatique
de modélisation et de leurs Applications

COMPLEXE DES CEZEAUX
BP 125 – 63173 AUBIERE CEDEX

INRA-Unité d'Agronomie

Equipe Fonctionnement et
Gestion de l'Ecosystème Prairial

DOMAINE DE CROUEL
234, Avenue du Brézet
63039 CLERMONT-FERRAND CEDEX

Rapport de projet 2ème année 2003-2004

Simulateur Prairie-Troupeau : Développement d'un module de défoliation contrôlée

Présenté par : Raphaël MARTIN
Benoît PEAN

Durée : du 16/10/03 au 20/03/04

Responsable ISIMA : Mme Christine FORCE
Responsables INRA : M Pascal CARRERE
Mlle Frédérique LOUAULT

Remerciements

Nous tenons à remercier tout d'abord Monsieur Pascal CARRERE ainsi que Mademoiselle Frédérique LOUAULT pour leurs explications, leurs conseils et leur patience.

Nous remercions aussi Madame Christine FORCE pour ses idées et autres éclaircissements qui nous ont permis d'avancer.

Nous tenons enfin à remercier Séverine PORTAL et Romain SEGUY pour leur travail de qualité.

Table des abréviations

API	: Applications Programming Interface
DefoCtrl	: Défoliation Contrôlée
DC	: Défoliation Contrôlée
IHM	: Interface Humain Machine
INRA	: Institut National de la Recherche Agronomique
JNI	: Java Native Interface
JVM	: Java Virtual Machine
ModVege	: Modèle Végétal
MVC	: Model / View / Controller
PARIS	: Plot And Ruminant Interaction Simulator
RS	: compartiment Reproducteur Sec (tiges et épis secs)
RV	: compartiment Reproducteur Vert (tiges et épis verts)
UML	: Unified Modeling Language – Langage de modélisation unifié
VS	: compartiment Végétatif Sec (gaines et feuilles sèches)
VV	: compartiment Végétatif Vert (gaines et feuilles vertes)

Glossaire

Apex	:	Partie de la plante à partir de laquelle se développent la tige et l'épi.
Abscission	:	Fonction qui détermine la quantité de matière sénescente qui sort du système chaque jour.
Biomasse	:	Quantité de matière végétale (fourrage) disponible sur une cellule.
Cellule	:	Une cellule est un ensemble de quatre compartiments qui représentent les états possibles de la matière végétale.
Cfg	:	Extension de fichier pouvant être édité par des éditeurs basiques et capable de supporter des lignes de commentaires.
Croissance	:	La fonction de croissance détermine la quantité de biomasse (g MS j^{-1}) qui sera produite chaque jour par le compartiment VV.
Csv	:	Extension de fichier exploitable par les tableurs type Excel.
Défolier	:	Action de prélever une quantité de biomasse d'une cellule
Faciès	:	Ensemble unique représentant une communauté végétale de quelques espèces dans des proportions fixées.
Litière	:	La fonction de litière calcule le flux d'abscission des compartiments sénescents.
Montaison	:	Fonction simulant le développement de la phase reproductrice des espèces végétales. Cette fonction permet de diriger une partie de la biomasse synthétisée dans le VV vers le RV.
Paquetage	:	Sous-système ou module regroupant un ensemble de classes fortement couplées [HILL 2003].
Sénescence	:	Fonction simulant les mécanismes de vieillissement naturel des tissus végétaux. La sénescence s'accompagne d'un changement d'état des organes qui se dessèchent et passent de « vert » à « sec ». Les fonctions de sénescence gèrent le transfert de la biomasse végétative verte vers le compartiment végétatif sec, et le passage de la biomasse reproductive verte vers le compartiment reproductif sec.
Station Alimentaire	:	Surface virtuellement exploitable par un animal sans qu'il bouge les pattes avant.

Table des figures et illustrations

Résumé

Dans le cadre du développement d'un **simulateur** modélisant l'interaction entre une prairie et le troupeau qui la pâture, l'Unité d'Agronomie de l'Inra nous a demandé de procéder à une amélioration du simulateur existant en modifiant deux de ses fonctions, la **montaison** et la **sénescence**. Ces modifications, qui se sont greffées sur le code **C++**, permettent d'améliorer la pertinence des sorties du simulateur tout en augmentant faiblement sa complexité grâce à l'utilisation du patron « **strategy** ».

Ensuite, nous avons élaboré le **module de défoliation contrôlée** qui permet de gérer des ablations déterministes de biomasse afin de modéliser la réponse de la végétation à ces prélèvements. Ce module a été développé en **C++** pour garder l'homogénéité du code avec le **simulateur**. Pour utiliser ce module de manière plus aisée, nous avons créé une **interface graphique**, basée sur celle développée par Séverine Portal et Romain Séguy lors d'un précédent projet, en suivant le patron **MVC**. Cette interface a été codée en **JAVA**, dans un souci d'homogénéité et de portabilité, en utilisant le logiciel JBuilder 7.0.

Mots clés : simulateur, montaison, sénescence, C++, strategy, module de défoliation contrôlée, interface graphique, MVC, JAVA

Abstract

Research workers and students have developed a **simulator** that modelled interaction between a plot and a ruminant herd. The Agronomical Unit of the INRA ask us to modify in **C++** language two functions, the **flowering** and the **senescence**, in order to make them more realistic. To do that, we decide to use the "**strategy**" design pattern, because it doesn't increase so much complexity of the **simulator**.

Then, we create a **controlled defoliation unit**, which enable to model plant reaction after known ablation of biomass. This unit was developed with **C++** language to keep the same structure in the **simulator** code. To use it easily, we create a **graphic interface**, based on the one that was created by Severine Portal and Romain Seguy in a former project. This interface was implemented in **JAVA** language using the "**MVC**" design pattern and the freeware JBuilder 7.0.

Key words: simulator, C++, flowering, senescence, strategy, controlled defoliation unit, graphic interface, MVC, JAVA

Table des matières

Introduction

Dans le cadre de notre deuxième année d'études à l'ISIMA, nous avons réalisé un projet de cent heures réparties sur quatre mois, entre novembre 2003 et mars 2004, pour le compte de l'Unité d'Agronomie de l'Institut National de Recherche Agronomique de Clermont-Ferrand.

Notre projet s'intègre dans un programme de recherche pluridisciplinaire intitulé « utilisation par le pâturage et l'évolution de la végétation des surfaces herbagères hétérogènes et peu chargées », qui a pour objectif de développer un modèle multi-agents spatialisé d'un troupeau de ruminants pâturant une prairie hétérogène. Le développement de cet outil de recherche, qui a été développé depuis plusieurs années, s'appuie sur une collaboration avec le LIMOS et l'ISIMA, et a intégré le travail de nombreux étudiants ingénieurs, à travers des projets et stages. Les finalités de ce programme sont de concevoir un outil de recherche qui permette de modéliser un système biologique complexe et hiérarchisé et d'étudier sa dynamique pluriannuelle. Les objectifs d'un tel programme sont de mieux comprendre le fonctionnement de cet écosystème et de voir les conséquences des différentes gestions qui sont appliquées sur son évolution. Ainsi, pour la végétation, différents scénarii pourront être testés pour considérer l'impact des pratiques d'élevage sur la dynamique prairiale, sa valeur d'usage et sa valeur environnementale, tout en évaluant les performances animales. Les applications seront trouvées dans la proposition d'itinéraires de gestion permettant de respecter à la fois un niveau de production économique satisfaisant et le respect de l'environnement (maintien de la biodiversité, ouverture du paysage, maintien de la fertilité des sols, etc.).

Ce simulateur a bien évolué depuis ses débuts et intègre de nombreuses fonctionnalités telles qu'une interface graphique, une visionneuse, un assistant d'analyse de sensibilité, etc.

Les premières étapes d'analyse de sensibilité et de vérification du prototype ont fait apparaître deux problèmes majeurs dans le simulateur. Le premier est un contrôle imparfait de la montaison par la défoliation animale. En effet, le passage du cycle végétatif au cycle reproducteur est contrôlé par l'action de l'animal en situation réelle, alors que dans le simulateur cela n'était qu'incomplètement pris en compte et conduisait à des dysfonctionnement. Le second problème réside en un décalage des flux de mortalité des tissus liés à la résolution mathématique des fonctions. Pour parer à ce problème il est nécessaire d'ajouter un descripteur « âge » des états de végétation permettant de pondérer ces flux de vieillissement et de les rendre plus conformes à ceux mesurés *in situ*.

Le simulateur dans sa fonction actuelle ne permet pas de contrôler le niveau de prélèvement par l'animal en un point donné de la prairie. Il est donc impossible de relier un état de végétation avec le prélèvement qu'elle a subi précédemment. Or il a été clairement montré que la défoliation animale en

fréquence et en intensité est un moteur déterminant de la dynamique de la végétation. C'est pour pouvoir proposer des règles d'évolution de la végétation en relation avec le niveau de perturbation (de prélèvement) subi que les chercheurs ont besoin actuellement d'un outil qui leur permette de défolier de manière totalement déterministe une cellule de végétation. En outre cet outil permettra de tester le comportement de la végétation en condition pâturée, chose qui n'est pas réalisée actuellement. A plus long terme, les résultats issus de cet outil permettront de proposer des règles d'évolution des états et des paramètres fonctionnels de la végétation qui seront intégrés dans la dynamique pluriannuelle des communautés végétales.

La demande qui nous a été formulée reprenait ces deux points. Notre travail a consisté dans un premier temps à implémenter ou modifier les deux fonctions incriminées : la montaison et la sénescence. Puis, dans un second temps nous avons développé à proprement parler le module de défoliation contrôlée ainsi que l'interface graphique qui l'accompagne.

Nous présenterons donc tout d'abord et succinctement la globalité du simulateur en approfondissant les points propres au sous-modèle végétal et à l'interface. Puis nous expliciterons les modifications apportées au sous-modèle Végétal. Nous expliquerons alors l'implémentation du module de défoliation contrôlée, qui sera suivie par celle de l'interface, pour finalement conclure et discuter des résultats.

Partie I : Contexte scientifique et technique

1.1 Le simulateur parcelle/troupeau

1.1.1 Description générale

Ce simulateur a pour objectif de modéliser le comportement d'un troupeau d'herbivores pâturant sur une parcelle de prairie hétérogène [BAUMONT et al. 2002]. Pour représenter les interactions entre animaux, la variabilité de leur comportement et l'utilisation spatiale d'une végétation hétérogène, une approche multi-agents spatialisée a été choisie [PEROCHON et al . 2001] . Ce troupeau est constitué d'un nombre quelconque d'animaux qui suivent tous un comportement aussi bien individuel que collectif; la croissance et le vieillissement de la végétation est elle aussi modélisée de manière valide [CARRERE et al.]. Tout cela implique un haut niveau de complexité et a amené les précédents programmeurs à choisir de décomposer ce simulateur en plusieurs sous-modèles qui seront chargés chacun de gérer un point précis.

On peut donc décompter pour l'instant trois principaux sous-modèles :

- le sous-modèle Animal qui est chargé de faire prendre au ruminant des décisions en fonction de son état physiologique.
- le sous-modèle Social qui est chargé de gérer les relations d'interactions et de compétitions au sein du troupeau. Il régit ainsi les déplacements longs et la distance entre les animaux.
- Le sous-modèle Végétal ou ModVege qui gère le fonctionnement de la végétation (montaison, sénescence) et sa distribution spatiale.

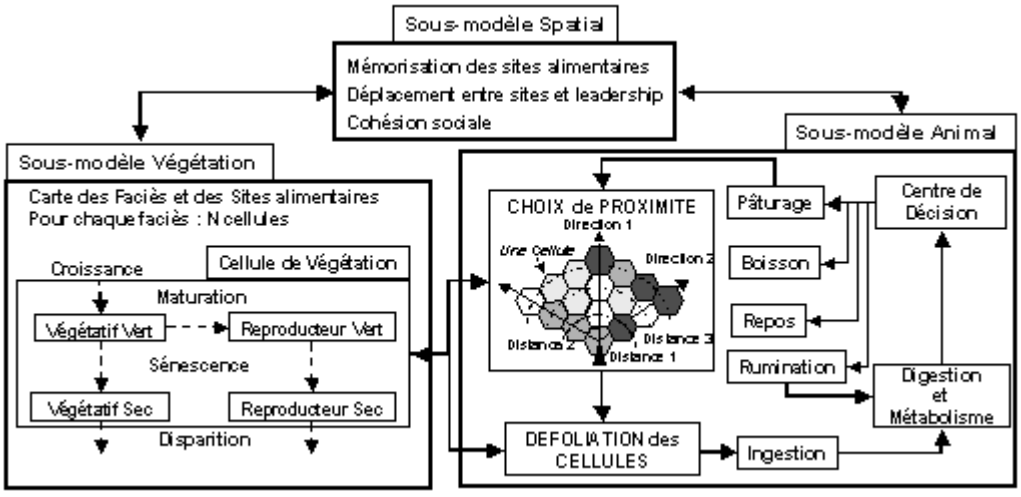

Figure 1 : Représentation simplifiée du simulateur [BAUMONT et coll. 2002]

Deux de ces sous-modèles (Animal et Végétal) ont été interconnectés lors du stage effectué par Guillaume MARTIN [MARTIN 2002] mais n'ont pour l'instant pas été validés.

1.1.2 Le Modèle Animal

1.1.2.1 Description générale

Le Modèle Animal permet de modéliser les réactions du ruminant en fonction de son état physiologique. C'est ce modèle qui permet à l'animal d'ingérer une partie de la végétation, cette dernière action modifiant ainsi son état physiologique. Ce modèle est principalement une simplification d'un autre développé précédemment [SAUVANT 1996 et al.].

Ainsi, la partie décisionnelle a été implémentée sous forme simplifiée. L'animal choisit alors de manger, de boire, de se reposer ou de se déplacer, ces actions modifiant donc ses motivations.

Ce modèle comporte aussi un générateur de trajectoire permettant à l'animal de se déplacer en fonction de la végétation alentour.

1.1.2.2 Entrées

Le Modèle Animal prend en entrée un premier fichier ruminant.csv qui permet de renseigner les différents paramètres concernant les animaux du troupeau. Il contient ainsi en première ligne le nom des paramètres puis, sur chaque ligne suivante, les valeurs correspondantes à un animal.

Le fichier troupeau.cfg comprend quant à lui tous les paramètres communs à tous les animaux du troupeau.

Le générateur de trajectoire utilise quant à lui un autre fichier .cfg qui contient les paramètres de configuration de la trajectoire des ruminants.

1.1.2.3 Sorties

Le Modèle Animal produit les fichiers suivants en sorties :

- Un fichier <nom_simulation><Jour>Perception<Preference>.trj décrivant la trajectoire du ruminant pour chaque jour de simulation.
- Un fichier <nom_simulation><Jour>Perception<Preference>.cel journalier permettant le visionnement de la parcelle grâce à la visionneuse.

- Un fichier sortieCentreDecisionnel.txt qui liste, entre autre, les différents choix d'activités de l'animal.
- Un fichier sortieRumen.txt qui liste les paramètres de l'animal à la fin de chaque séquence.
- Un fichier sortieRumenDetail.txt qui liste les paramètres de la classe RumenDetail de l'animal à chaque fin de séquence.
- Un fichier sortieIngestion.txt qui donne les différentes cellules ingérées par l'animal ainsi que les quantités prélevées sur ces cellules.

1.1.3 Le Modèle Social

Ce modèle gère les relations sociales et spatiales au sein du troupeau. Deux types de relations sociales vont influencer la position et le déplacement des animaux : le grégarisme et le leadership.

Dans un troupeau, les animaux ont tendance à rester groupés : c'est ce qu'on nomme le grégarisme. De plus, ils ont tendances à suivre un animal particulier du troupeau, appelé leader. A un moment donné, c'est l'animal leader qui peut provoquer des déplacements de groupe, appelés déplacements longs, pour se rendre sur un autre site de la parcelle afin de manger, boire ou se reposer puisqu'il est capable de mémoriser les sites précédemment visités.

En effet, chaque animal possède une mémoire pour enregistrer les meilleurs sites alimentaires ainsi que les sites de buvée et de repos. Or, de par le leadership, seul le leader peut consulter sa mémoire. Ainsi, lorsque les cellules voisines ne conviennent pas à ce dernier, il utilise alors sa mémoire dans le but de trouver un site plus approprié : ce sont les déplacements longs [PORTAL 2003].

1.1.4 Le Modèle Végétal

1.1.4.1 Description générale

Ce sous-modèle a trois objectifs principaux :
- la mise à jour de la parcelle, via la sénescence et la montaison de la végétation. La croissance des cellules tient compte du climat.
- la mise à disposition d'informations destinées au modèle Animal lors du choix par le ruminant de la cellule à défolier.
- la mise à jour, suite à la défoliation d'un animal, de la quantité de biomasse disponible dans la cellule.

Il simule le fonctionnement d'un couvert prairial plurispécifique, défini par des communautés végétales appelées faciès. Chaque faciès est défini par 25

paramètres qui spécifient entre autres, le seuil de démarrage de la végétation, sa précocité d'épiaison, sa digestibilité ou la vitesse d'abscission des tissus morts. L'unité de base de la végétation est une cellule hexagonale sur laquelle toutes les fonctions, dont la défoliation, sont appliquées de façon homogène. La taille de la cellule est actuellement paramétrée à 0.1 m^2, ce qui correspond à la taille d'une station alimentaire chez le mouton. Dans chaque cellule la végétation épigée est décrite par quatre compartiments : gaines et feuilles vertes (végétatif vert), gaines et feuilles sèches (végétatif sec), tiges et épis verts (reproducteur vert), tiges et épis secs (reproducteur sec). Chaque compartiment est décrit par des variables d'état (biomasse et hauteur) et des variables qualitatives (teneur en parois végétales (NDF) et digestibilité du NDF). Les interactions entre les compartiments sont simulées à partir de 6 fonctions qui calculent les flux de biomasses [CARRERE *et al*, 2002] : une fonction de croissance, une fonction de maturation (développement reproducteur), deux fonctions de sénescence des compartiments végétatifs et reproducteurs respectivement, et deux fonctions de disparition des tissus morts (Figure 2). A l'échelle de la parcelle, un fichier « carte » gère la répartition spatiale de la végétation. Chaque cellule est associée à un faciès qui renseigne les valeurs de ses paramètres et les cellules adjacentes d'un même faciès sont virtuellement regroupées pour définir des sites alimentaires.

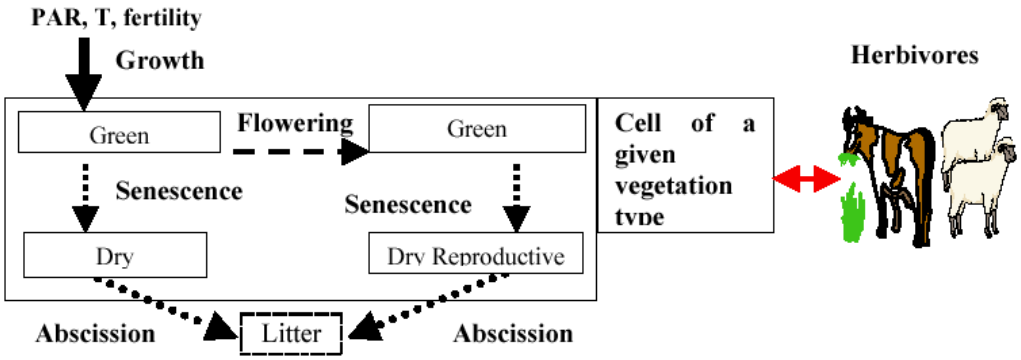

Figure 2 : Structure générale du sous-modèle Végétal [CARRERE et al. 2002]

1.1.4.2 Entrées

Tous les fichiers d'entrées se trouvent dans le répertoire Input et ont l'extension .csv.

Le fichier param.csv contient toutes les informations utiles au bon fonctionnement global du modèle. Il contient des informations telles que les dimensions de la parcelle ou encore le nombre de faciès.

Le fichier constant.csv contient les constantes nécessaires au fonctionnement du Modèle Végétal :

- S : surface de la cellule en m^2.

- RUEmax : efficience de conversion de la lumière maximale.
- INO : indice de nutrition minimale.
- k : coefficient d'extinction.
- σ : perte de matière par respiration lors de la sénescence du compartiment végétatif.
- σ' : perte de matière par respiration lors de la sénescence du compartiment reproducteur.

Le fichier environnement.csv définit pour chaque jour la température moyenne en degrés Celsius ainsi que le rayonnement photosynthétique actif incident ou PARi en MJ. m^{-2} .j^{-1}.

Le fichier facies.csv contient les différentes informations relatives à chaque faciès (43 au total).

Le fichier site.csv permet d'associer un numéro de site à un unique numéro de faciès. Un faciès peut quant à lui être associé à plusieurs sites.

Le fichier carte.csv représente la parcelle cellule par cellule, chaque numéro correspondant au site associé à la cellule. Ce fichier peut-être généré par le créateur.

1.1.4.3 Sorties

Le Modèle Végétal produit les fichiers suivants :
- choix.csv et choixTemp.csv qui décrivent le choix des cellules.
- defol.csv et defolTemp.csv qui décrivent la défoliation des cellules.
- Maj_jour.csv décrit l'état des cellules à la fin du jour, où jour est le numéro du jour dans la simulation.

1.2 Arborescence utilisée par le simulateur

Deux arborescences différentes ont été proposées lors de deux projets. Nous avons décidé de calquer la nôtre sur celle mise en place par Séverine PORTAL et Romain SEGUY [PORTAL et SEGUY 2003] car elle permet de structurer clairement les fichiers d'entrées et sorties sous divers répertoires. Voici cette arborescence :

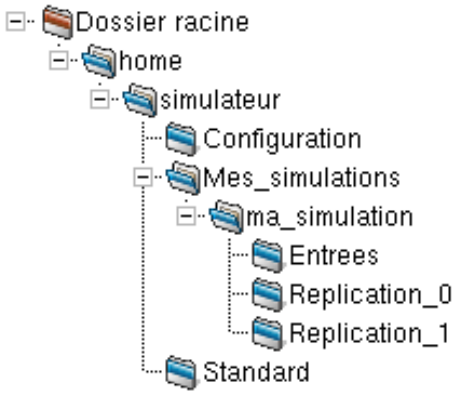

Figure 3 : Arborescence « type »

Le répertoire simulateur se trouve sous /home dans une optique multi-utilisateur du simulateur PARIS.

Le sous-répertoire Configuration contient les fichiers d'entrées utiles à l'interface.

Le sous-répertoire Mes_simulations contient, comme son nom l'indique, l'ensemble des simulations. Dans la figure ci-dessus, on peut constater la présence d'une simulation, nommée ma_simulation, qui contient un répertoire comportant les divers fichiers d'entrées ainsi qu'un répertoire pour chaque réplication réalisée. On dit qu'il y a réplication quand on réalise plusieurs fois la même simulation du point de vue des entrées mais en utilisant un germe différent pour le générateur de nombres pseudo aléatoires. C'est dans ces répertoires que l'on trouve tous les fichiers de sorties des différents modèles.

Enfin, le répertoire standard contient tous les fichiers permettant de lancer une simulation avec les paramètres par défaut.

1.3 L'Interface

1.3.1 Présentation

Cette interface a été développée par Séverine PORTAL et Romain SEGUY [PORTAL et SEGUY 2003] à l'occasion du projet effectué lors de leur seconde année d'ISIMA. Ainsi cette partie présentera, dans un degré moindre que leur rapport, l'essentiel de leur travail.

L'interface est structurée en trois parties bien distinctes :

- La première partie, située sur la gauche de l'interface, permet à l'utilisateur d'accéder, à tout instant, au sous-système et à l'étape de son choix. Elle indique aussi l'état de chaque étape, en utilisant un code des

couleurs : une étape peut être validée (couleur verte), en cours de validation (couleur orange) ou bien non encore validée (couleur rouge).

- La deuxième partie, située dans la zone supérieure droite de l'interface, présente à l'utilisateur les composants graphiques représentant l'étape choisie à un instant donné. C'est dans cette zone que l'utilisateur effectue les actions permettant de valider une étape et donc de paramétrer le simulateur.
- La dernière partie de l'interface est la zone d'aide contextuelle, située dans la zone inférieure droite de l'interface. A chaque étape correspond une aide succincte mais précise guidant l'utilisateur en cas de doutes.

Pour permettre la navigation entre les étapes, deux boutons « Précédent » et « Suivant » sont disposés en bas de l'interface, ainsi qu'un bouton « Lancer la simulation » permettant, comme son libellé l'indique, de lancer la simulation, une fois toutes les étapes validées.

La figure 4 présente l'écran principal de l'interface tel qu'il pourrait se présenter lors d'une étape :

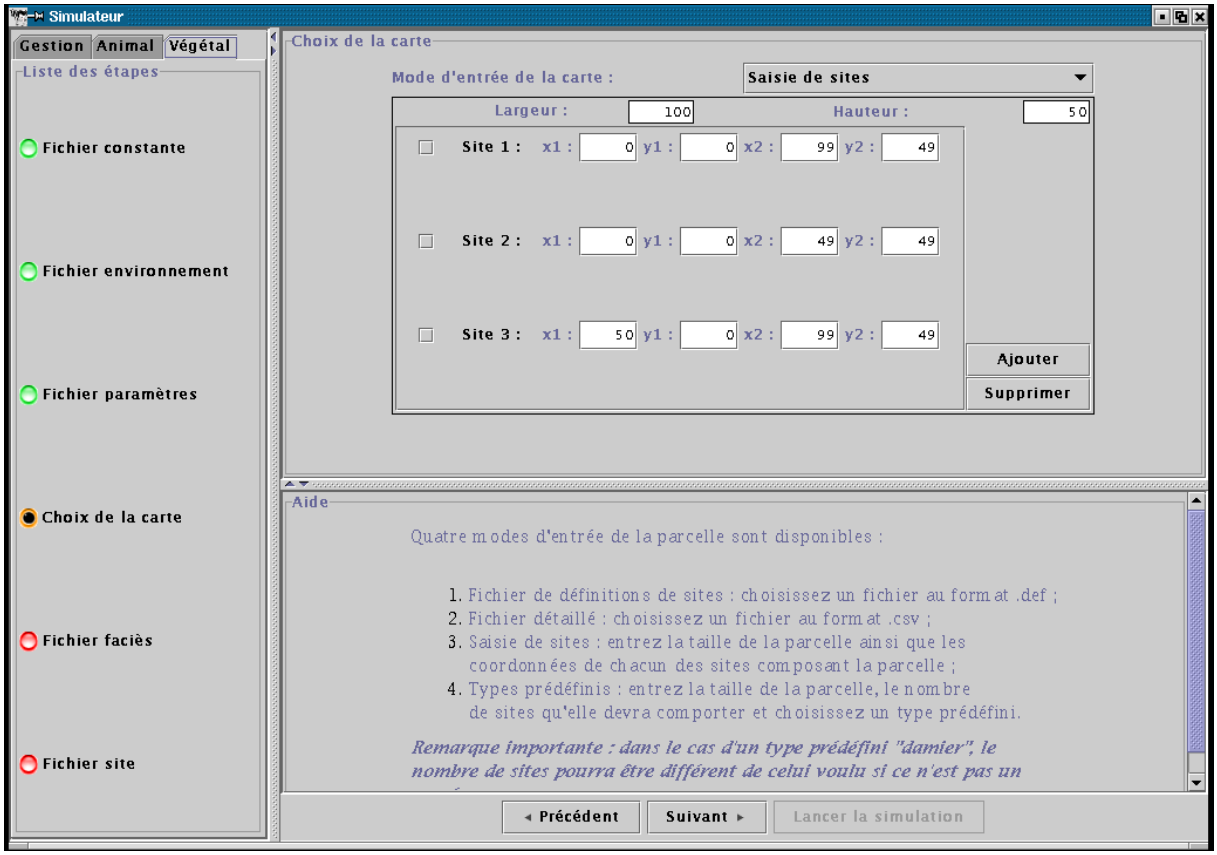

Figure 4 : Ecran principal de l'interface

1.3.2 Solution de conception retenue

1.3.2.1 Le patron MVC

L'interface a été réalisée en utilisant le patron MVC [PORTAL et SEGUY 2003]. Ce choix se justifie par le fait que ce patron a été de nombreuses fois testé et approuvé. De plus, le code généré est intelligible et permet de dissocier la partie purement graphique des traitements comme le générateur de carte. L'interface est divisée en trois parties :

- la partie de l'interface utilisateur qui affiche des informations à propos du modèle, appelée « vue ».
- la partie de l'interface utilisateur qui permet d'agir sur le modèle, nommée « contrôleur ».
- le « modèle », ou « métier », qui constitue le cœur de l'application et contient les données et les traitements. Lorsque des changements interviennent au niveau du modèle, la vue est mise à jour par celui-ci

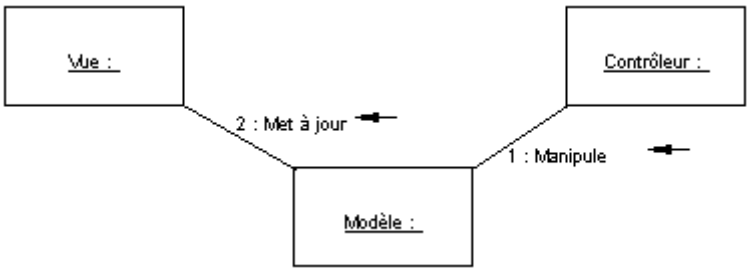

Figure 5 : Diagramme de collaboration du patron MVC

Toutefois, dans leur implémentation de ce patron, Séverine PORTAL et Romain SEGUY ont décidé de lier le contrôleur et la vue dans un but de simplification du code généré.

Ainsi, les classes de l'interface correspondant aux vues et aux contrôles se nomment toutes Vue suffixée d'un nom décrivant l'étape à valider par le biais de cette vue ; ces classes dérivent toutes de la classe abstraite VueEtape.

Les classes de l'interface correspondant aux modèles se nomment toutes Etape, suivi du nom décrivant l'étape et héritent toutes de la classe abstraite Etape.

A chaque vue correspond une unique étape : par exemple, à l'étape EtapeFichierCarte correspond la vue VueFichierCarte.

1.3.2.2 Les paquetages

L'interface a été décomposée selon les paquetages suivants :

- le paquetage fr.inra.clermont.simulateur regroupe les classes principales de l'interface et toutes les classes qui ne sont pas susceptibles de figurer dans les autres paquetages décrits ci-dessous et qui ne justifient pas la création d'un paquetage particulier.
- le paquetage fr.inra.clermont.simulateur.configuration regroupe toutes les classes nécessaires au paramétrage de l'interface.
- le paquetage fr.inra.clermont.simulateur.ihm regroupe toutes les classes graphiques de l'interface : vues, gestionnaire de vues, etc. Ce paquetage inclut aussi les classes permettant aux différents composants graphiques de l'interface de communiquer entre eux.
- le paquetage fr.inra.clermont.simulateur.operation regroupe toutes les classes correspondant aux différentes étapes.

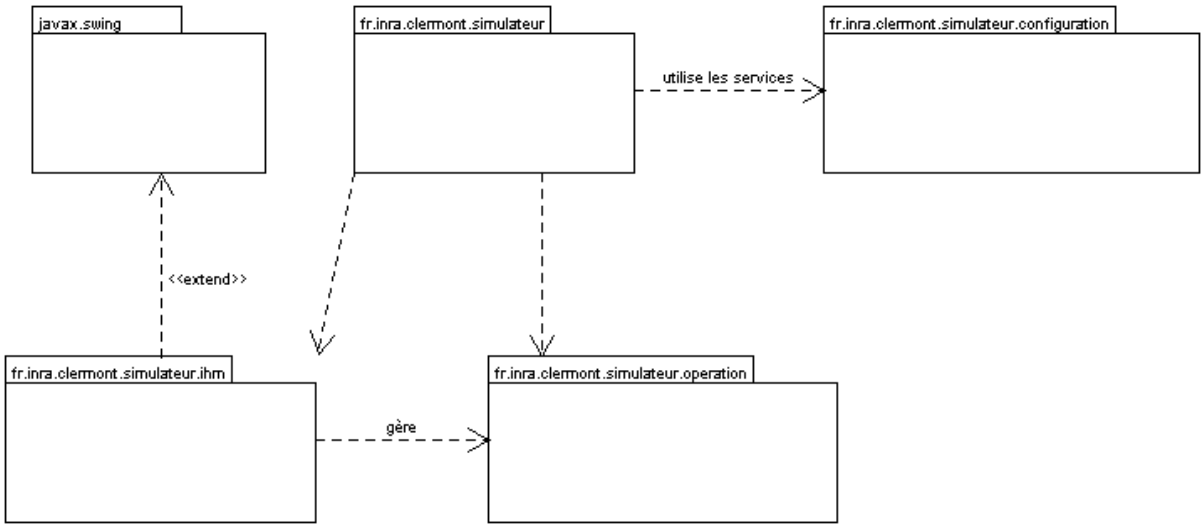

Figure 6 : Diagramme de paquetages de l'interface

1.3.3 Paquetage fr.inra.clermont.simulateur

Le but de ce paquetage est d'abord de fournir les classes permettant de lancer l'interface, l'interface utilisant principalement les services des autres paquetages. Ce paquetage est aussi utilisé pour contenir toutes les classes ne justifiant pas la création d'un paquetage particulier. Il comprend ainsi la classe principale de l'application, à savoir Interface.

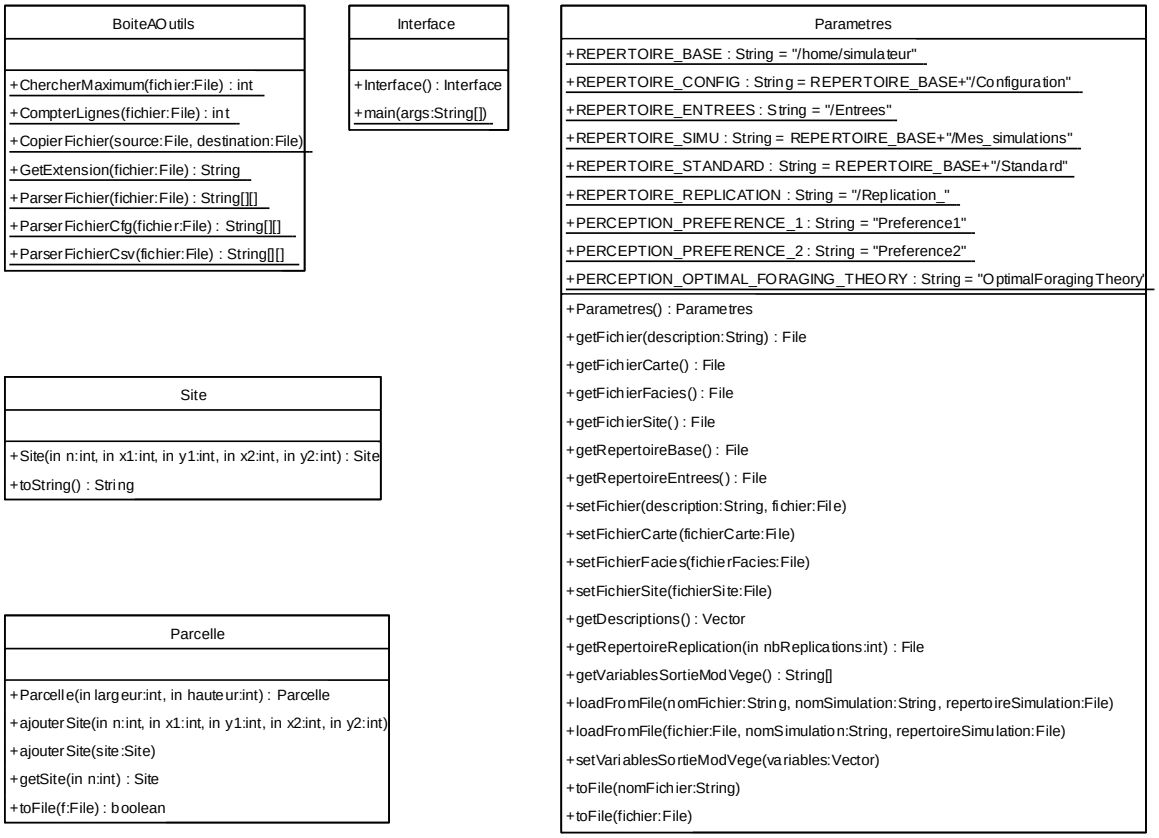

Figure 7 : Diagramme de classe du paquetage fr.inra.clermont.simulateur

1.3.4 Paquetage fr.inra.clermont.simulateur.configuration

Ce paquetage contient une unique classe Configuration qui permet d'instancier les étapes qui sont générées dynamiquement.

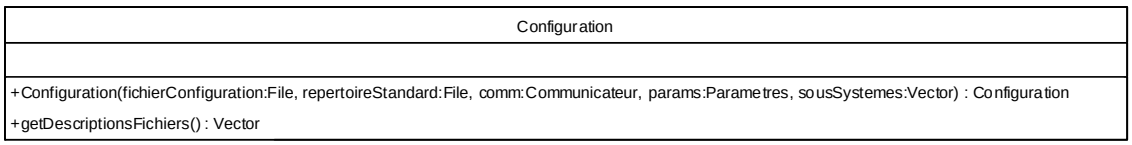

Figure 8 : Diagramme de classe du paquetage fr.inra.clermont.simulateur.configuration

1.3.5 Paquetage fr.inra.clermont.simulateur.operation

Ce paquetage a été créé pour contenir toutes les étapes et opérations menant à l'exécution d'une simulation.

Les deux classes essentielles de ce paquetage sont les classes Etape et EtapeFinale. Ces deux classes sont toutes les deux abstraites et sont nécessaires pour créer une étape quelconque.

La classe Etape représente une étape standard de l'interface. Une telle étape se définit par son niveau de priorité et son état.

La classe EtapeFinale représente un type d'étape particulier. En effet, elle ne correspond à aucune vue car elle correspond à l'étape exécutée lors du clic sur le bouton Lancer la simulation de l'interface.

Les étapes héritant directement des classes Etape et EtapeFinale sont considérées comme des étapes statiques. En effet, les étapes statiques sont des étapes dont le comportement ne peut pas être spécifié simplement dans un fichier de configuration, la gamme des opérations pouvant être effectuées étant trop vaste.

Les étapes statiques héritent donc directement de la classe Etape ou de la classe EtapeFinale et sont intégrées au sein de l'interface. Pour modifier les paramètres d'une étape statique, il est donc nécessaire de modifier le code source de l'application, d'où l'aspect statique de ces étapes.

De plus, l'interface doit être capable de gérer en entrée un nombre quelconque de fichiers de configuration, ceux-ci contenant eux-mêmes un nombre quelconque de paramètres.

L'interface doit donc pouvoir générer automatiquement les étapes et les vues associées à ces fichiers de configuration variables. Ces étapes sont donc dynamiques, dans le sens où elles sont créées lors du lancement de l'interface, par opposition aux étapes statiques qui sont programmées directement dans l'application.

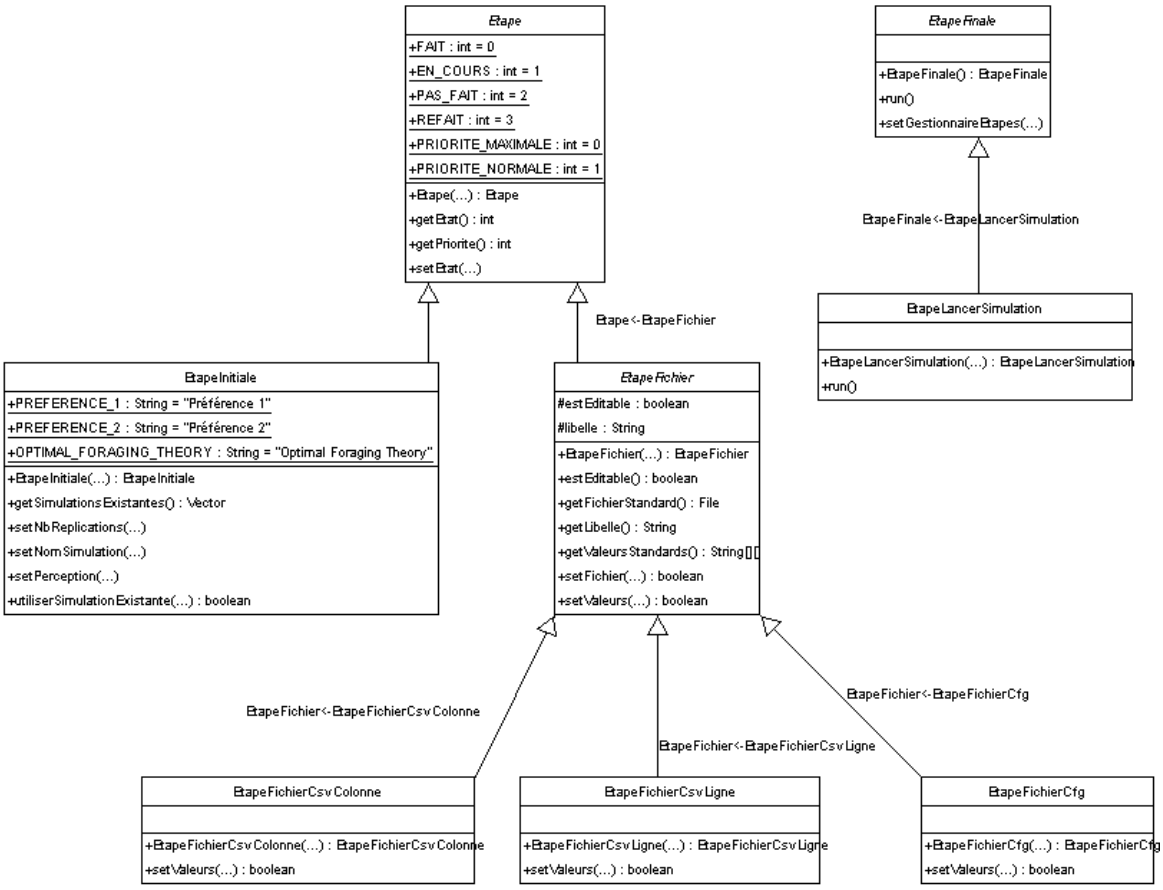

Figure 9 : Diagramme de classe du paquetage fr.inra.clermont.simulateur.operation

1.3.6 Paquetage fr.inra.clermont.simulateur.ihm

Le paquetage fr.inra.clermont.simulateur.ihm peut être considéré comme le paquetage central de l'interface. En effet, il contient l'intégralité des classes permettant de présenter à l'utilisateur une interface graphique évoluée.

La classe la plus importante de ce paquetage est sans doute la classe GestionnaireEtapes. Le gestionnaire d'étapes désigne la fenêtre chargée d'héberger les vues correspondant aux étapes. C'est lui qui orchestre l'enchaînement des étapes : c'est au sein cette classe qu'est implémenté le mécanisme de validation des étapes. Enfin, le gestionnaire d'étapes prend en compte les évènements déclenchés par l'utilisateur.

Comme dans le paquetage fr.inra.clermont.simulateur.operation, des « vues » ont dû être gérées dynamiquement, d'où la présence des classes VueFichierCfg, VueFichierCsvColonne et VueFichierCsvLigne.

La dernière classe importante du paquetage simulateur.ihm est la classe Communicateur. Un communicateur désigne un unique objet commun à toutes les vues leur permettant d'échanger des messages.

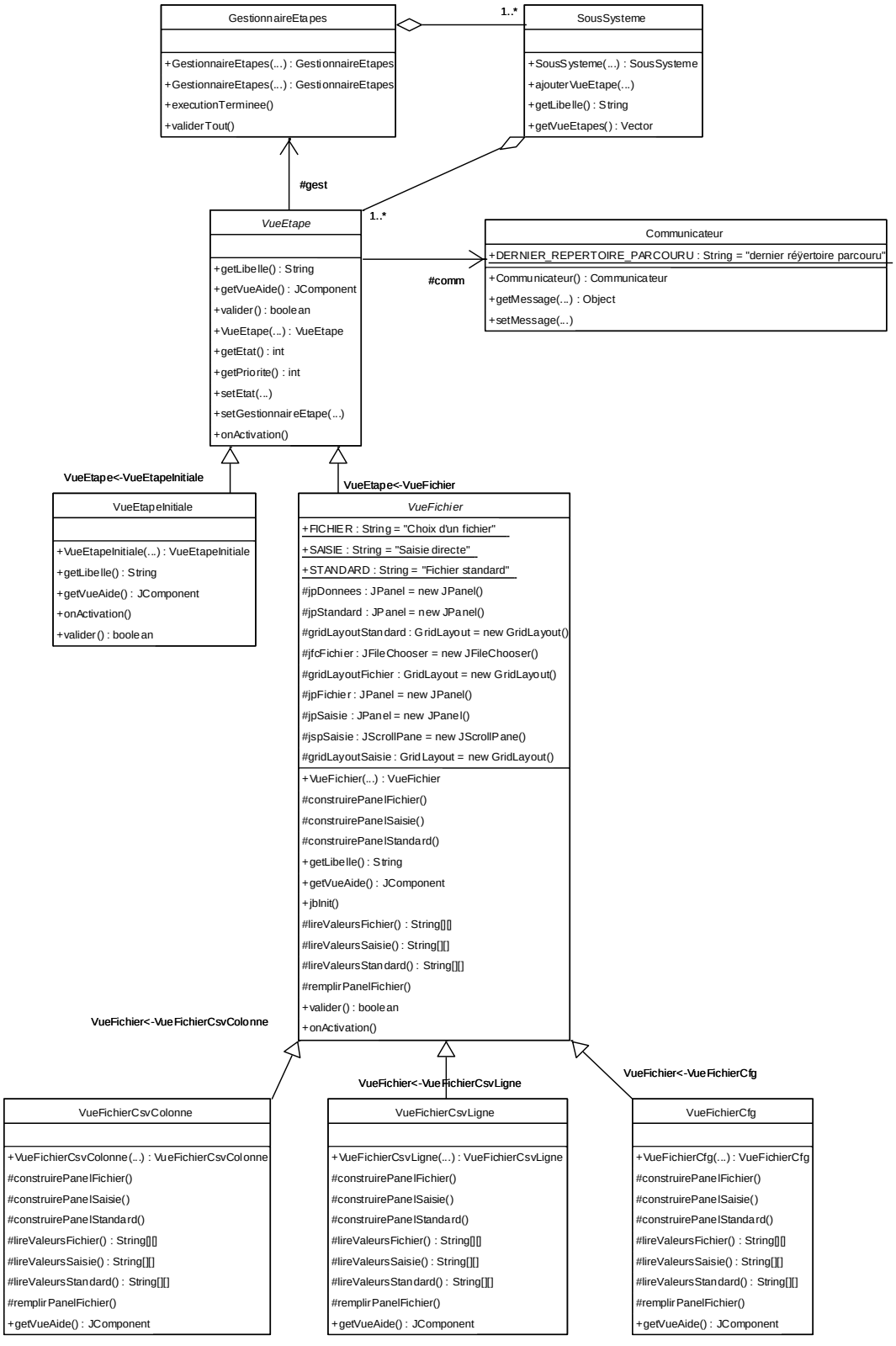

Figure 10 : Diagramme de classe du paquetage fr.inra.clermont.simulateur.ihm

Partie 2 : Modifications apportées au Modèle Végétal

Dans un premier temps nous avons modifié les fonctions biologiques du modèle végétal pour améliorer le pouvoir de simulation du modèle et rendre ses sorties plus proches de résultats expérimentaux obtenus sur le terrain. Les fonctions modifiées sont la fonction de montaison et les fonctions de sénescence et de litière.

2.1 Fonction de montaison

2.1.1 Description de la fonction

La fonction de montaison répartit entre les compartiments VV et RV la biomasse journalière produite selon un coefficient d'allocation déterminé par la fertilité du milieu. Elle permet de modéliser le développement des organes reproducteurs. Elle s'effectue lors de la mise à jour du compartiment mais ne se déclenche qu'entre deux dates, qui correspondent au début et à la fin de la période reproductive. La gestion de ces dates de lancement et de fin est réalisée par un compteur qui cumule les températures moyennes journalières (somme des températures) depuis le début de la simulation.

2.1.2 Nécessité de modifier cette fonction

Cela induit le fait que tant qu'il y aura production de biomasse, il y aura, entre les deux dates de lancement et de fin de la montaison, création de matériel reproducteur. Cela pose le problème de l'impossibilité de contrôler la production d'épis par la défoliation animale. Or ce contrôle est réalisé par la conduite du troupeau en situation réelle et doit pouvoir être modélisé. Cela est indispensable pour simuler le fonctionnement de la végétation pâturée, et c'est d'autant plus important que le pâturage animal est intensif. Les modifications que nous avons effectuées ont pour objectif de rendre possible ce contrôle.

2.1.3 Solutions choisies

Il faut introduire une règle de contrôle supplémentaire de la fonction de montaison.

En situation de pâturage, la limitation du développement reproducteur se fait au moyen d'interventions (déprimage, étêtage) visant à supprimer les apex floraux. Les solutions intégrées dans le modèle reprennent cette logique de gestion, en cherchant à rédiger des règles visant à contrôler l'ablation des apex. Trois règles sont proposées, dans l'attente d'une validation ultérieure.

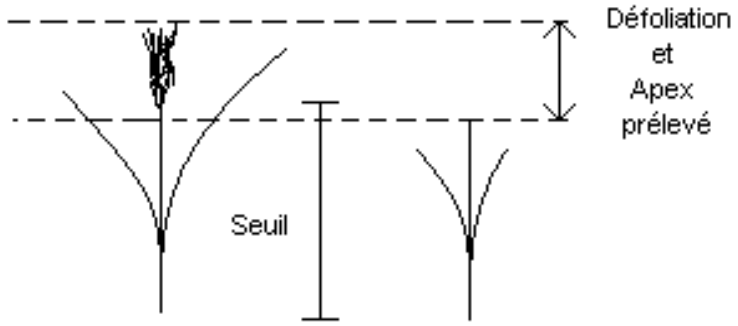

Figure 11 : Illustration de l'objectif des modifications de la fonction montaison

- *1ere solution :*

  On considère que le stade critique du prélèvement des apex est déterminé par le stade « épi 10 cm », c'est-à-dire le stade pour lequel la hauteur de l'apex floral est à 10 cm au-dessus du sol. On reprend cet indicateur et on réalise un test : si la hauteur du compartiment RV avant défoliation est supérieure à 10 cm (apex accessibles) et si elle est inférieure après défoliation, alors l'apex est prélevé et la montaison s'arrête.

- *2eme solution :*

  C'est le même principe que ci-dessus mais le test ne s'effectue pas sur la valeur de 10 cm mais sur Hcell qui est la moyenne pondérée des hauteurs de tous les compartiments. Donc, si la hauteur du compartiment RV avant défoliation est supérieure ou égale à la hauteur moyenne de la cellule avant défoliation (apex accessibles), et que la hauteur du compartiment RV après défoliation est inférieure ou égale à la hauteur moyenne de la végétation avant défoliation, alors l'apex est prélevé et la montaison s'arrête.

- *3e solution :*

  On considère ici que sur une cellule de 0.1 m², tous les apex ne sont pas forcément prélevés, et que la proportion des apex prélevés sera proportionnelle à l'intensité du prélèvement animal. L'intensité du prélèvement se définie comme la quantité de biomasse défoliée par rapport à la quantité de biomasse offerte (présente sur la cellule). Cela signifie que pour une défoliation légère, une proportion faible des apex est prélevée, alors que pour une défoliation plus importante la quasi-totalité

des apex est prélevée. Nous établissons donc une fonction qui reliera la fonction de montaison à l'intensité de défoliation.

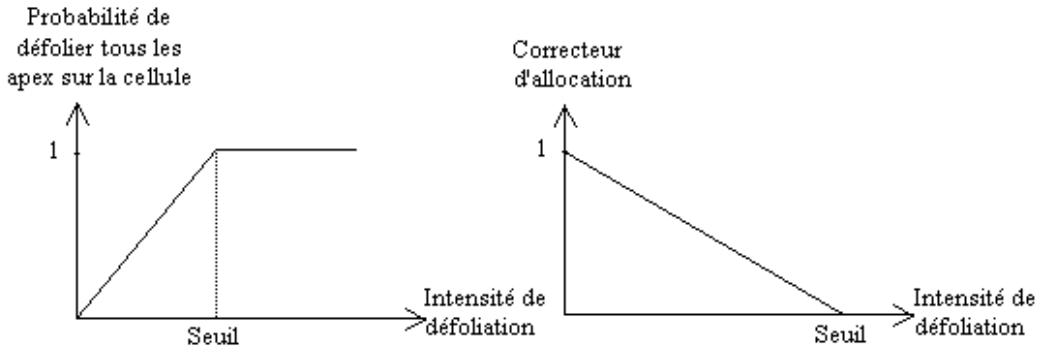

Figure 12 : Graphique illustrant la troisième solution de la fonction de montaison

La valeur du coefficient d'allocation sera modifiée par un correcteur d'allocation déterminé par la fonction schématisée à la Figure 12.

Le seuil d'intensité sera paramétré autour de 50% et sera fixé dans le fichier constante.csv.

Ces trois solutions permettront de tester l'impact d'une loi de contrôle de la montaison de type tout ou rien dans les solutions 1 et 2 (la solution 1 n'étant qu'un cas particulier de la solution 2), et l'impact d'une loi continue dans le cas de la solution 3.

2.1.4 Implémentation

Dans l'attente d'une validation, les trois solutions ont été implémentées. Pour ce faire, nous avons utilisé le patron de conception «strategy». Ce patron permet de définir une famille d'algorithmes et les rend interchangeables : les algorithmes peuvent varier indépendamment de l'utilisateur.

Sa structure est composée d'une classe Contexte, d'une classe Stratégie et de classes Stratégie concrète.

Les classes Stratégie concrète héritent de la classe abstraite Stratégie qui déclare l'interface commune à tous les algorithmes.

Ici la classe FctMontaison, dont la méthode montaison est virtuelle pure, est la stratégie et la classe CompartimentVV le contexte. Les classes FctMontaison1, 2 et 3 correspondent respectivement aux trois solutions exposées

précédemment. Le choix de la stratégie sera fixé dans le fichier « constante » avec la valeur 1, 2 ou 3.

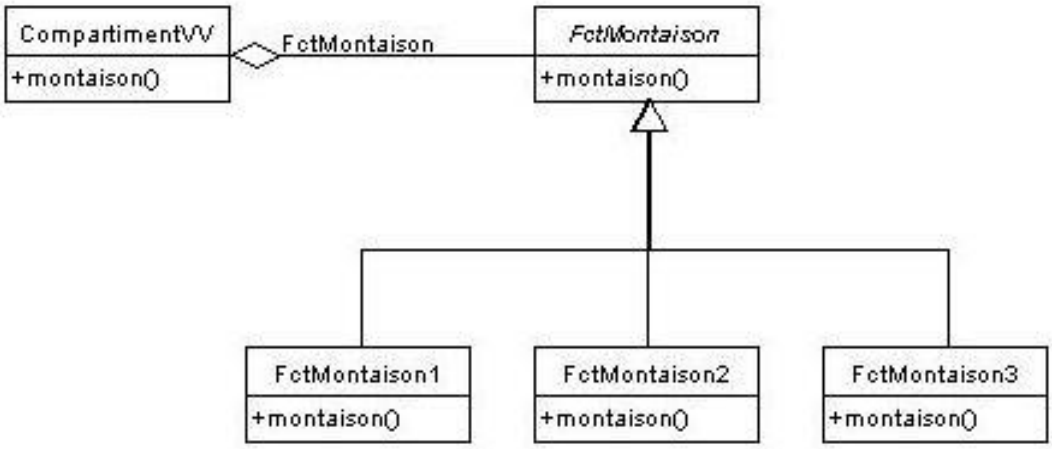

Figure 13 : Diagramme de classes du patron "strategy"

L'utilisation de ce patron de conception présente l'avantage d'éliminer les conditions et de regrouper différents algorithmes effectuant la même action. En revanche, il augmente le nombre d'objets et oblige l'utilisateur à comprendre comment les différentes stratégies fonctionnent afin de choisir celle convenant le mieux.

2.2 Fonctions de sénescence et de litière

2.2.1 Description des fonctions

La fonction de sénescence détermine la quantité de biomasse qui va passer d'un compartiment « Vert » à un compartiment « Sec ». Cette fonction tient compte de la biomasse du compartiment vert. Deux fonctions de sénescence sont implémentées, l'une gérant la sénescence des tissus végétatifs (de VV vers VS), l'autre celle des tissus reproductifs (de RV vers RS). La quantité journalière de biomasse résultant du calcul de la sénescence viendra augmenter la biomasse des compartiments VS et RS et sera déduite respectivement des compartiments VV et RV.

La fonction de litière calcule le flux d'abscission des compartiments sénescents, c'est à dire détermine la quantité de matériel sénescent qui va sortir du

système chaque jour. Cette fonction tient compte de la biomasse des compartiments « Sec ». Deux fonctions litière sont modélisées, l'une s'appliquant au compartiment VS, l'autre au compartiment RS. L'influence du climat sur l'abscission des tissus sénescents se fera en exprimant les coefficients de dégradation du matériel sénescent en degrés jour plutôt qu'en jour.

2.2.2 Nécessité de modifier ces fonctions

La sénescence et l'abscission se font avec un temps de décalage par rapport au flux de croissance. Mais comme les flux de sénescence et d'abscission sont fonction de la température moyenne journalière, le maximum de biomasse, obtenu pendant une période de températures élevées, sera à évacuer pendant une période de températures plus faibles. Il y a donc accumulation de matière sèche des compartiments VS et RS.

On souhaite donc "forcer" la sénescence et l'abscission de façon à être plus proche de la "digestion hivernale" du sénescent observé expérimentalement.

2.2.3 Solution choisie et implémentation

Pour effectuer ce forçage, on introduit un coefficient de forçage en fonction de l'âge moyen du compartiment. Cela nécessite donc d'ajouter un attribut Age_X pour chaque compartiment X(VV, RV, VS, RS) qui sera recalculé à chaque mise à jour de la biomasse. L'âge moyen d'un compartiment est l'âge pondéré en fonction des entrées et des sorties de biomasse dans ce compartiment. Il se calcule selon la formule suivante :

$$\mathbf{Age\_X_t = [(W X_{t-1} – S W_X) * Age\_X_{t-1} + 1] / W X_t}$$

Où :
Age_X_t est l'âge moyen du compartiment au jour t.
WX_{t-1} est la biomasse du compartiment au jour précédent (t-1)
SW_X est le bilan des sorties du compartiment
Age_X_{t-1} est l'age moyen du compartiment au jour précédent +1 car la biomasse présente a vieilli d'un jour.
WX_t est la biomasse du compartiment mise à jour (existante + entrées – sorties).

Les coefficients de forçage sont intégrés dans les équations de sénescence et de litière pour VS et RS : les coefficients de passage en sénescence KX et en litière KlX présents dans ces équations deviendront KX * fX(age) et KlX * gX(age) où f et g sont des fonctions permettant de déterminer les coefficients de forçage et X représente le compartiment.

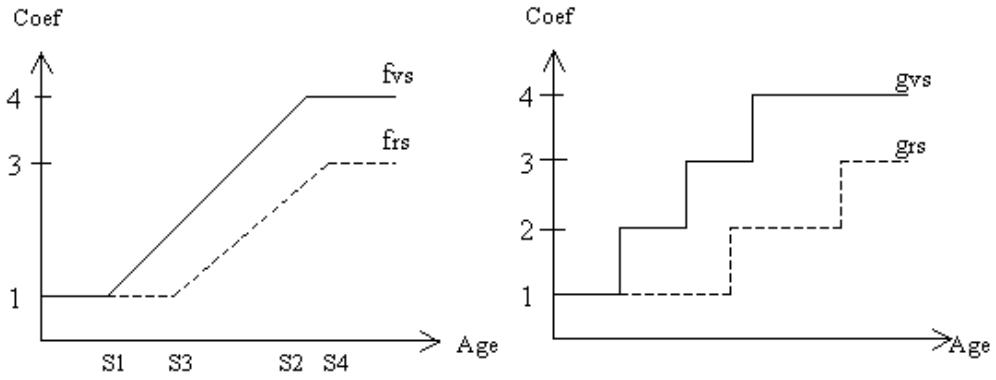

Figure 14 : Fonctions donnant les coefficients de forçage

Exemple de valeurs seuils définies par défaut (en attente de validation) :

VS	S1 : 20 j	S2 : 80j	VS	seuils : 20	g(age) =	2
RS	S3 : 40j	S4 : 90j		40		3
				60		4
			RS	seuils : 50		2
				100		3

Ces valeurs ne sont données qu'à titre indicatif et seront fixées dans le fichier constante.csv.

Partie III : Module de défoliation contrôlée et Interface graphique associée

3.1 Cahier des charges

3.1.1 Objectif

Ce module a pour objectif de simuler de manière totalement déterministe la défoliation d'une cellule. Il ne tient pas compte des paramètres comportementaux des animaux (déplacement, sélection …). On ne s'intéresse qu'au point de vue de la végétation, pour laquelle la défoliation se traduit par une ablation de tissus. Ce module ne travaille que sur une seule cellule et va permettre d'analyser la réponse de cette cellule à différentes défoliations.

3.1.2 Démarche

On va considérer deux priorités.

3.1.2.1 Priorité1 : Description des paramètres de contrôle

Le module doit prendre en compte les deux composantes de la défoliation, à savoir l'intensité et la fréquence, et permettre de contrôler la composition du prélèvement en déterminant la quantité défoliée sur chaque compartiment de la cellule.

3.1.2.1.1 Fréquence des prélèvements

La fréquence définit l'intervalle entre deux défoliations successives, c'est-à-dire du point de vue de la végétation, détermine la durée de la repousse. Elle est exprimée soit en « jour, j » soit en « degrés.jours, °Cj » (somme des températures journalières positives).

Les paramètres de gestion de la fréquence sont :

- une date de départ, à laquelle s'effectue la première défoliation
- un intervalle qui est la fréquence choisie (mêmes unités que ci dessus)
- une date de fin et un nombre de défoliation (première défoliation incluse)

Trois options ont été retenues:

- Gestion calendaire
  La date de départ est définie par le numéro de jour calendaire. On comptabilise à partir du 1er janvier de l'année de référence; en cas de simulation sur plus d'un an, le 1er janvier de l'année n+1 a pour numéro 366.
  Le pas de temps est exprimé en jour (entier) et varie de 1 à 365.

- Gestion thermique
  La date de départ est définie par la somme des températures positives depuis le 1er janvier de l'année n de référence. En cas de simulation sur plus d'un an, on continue l'incrémentation en considérant le 1er janvier de l'année n comme point de départ.
  Le pas de temps est exprimé en degrés.jours, les jours étant entiers et variant de 1 à 365.

- Gestion sur les états
  La date de départ est définie par un état de biomasse.
  Le pas de temps est exprimé en fonction de la variable d'état.

Dans les trois cas, on introduit une date de fin exprimée en jour, et un nombre entier de défoliations. La période de défoliation s'arrête lorsque le premier des deux paramètres est atteint.

3.1.2.1.2 Intensité de défoliation

L'intensité est la proportion de l'offre herbagère qui est prélevée sur la cellule. Du point de vue végétal, elle détermine la biomasse résiduelle à partir de laquelle se fera la repousse. L'intensité sera gérée soit par la hauteur, soit par la biomasse.

- Gestion par la hauteur
  L'intensité est exprimée en pourcentage de la hauteur moyenne de la cellule (Hcell). Le prélèvement sera borné entre 0 et X%, où X sera déterminé de manière à ce qu'il reste toujours 1 cm (strate non pâturable).

- Gestion par la biomasse
  L'intensité du prélèvement est exprimée en pourcentage de la biomasse totale de la cellule. Le prélèvement sera borné entre 0 et X%, où X sera déterminé de manière à ce qu'il reste toujours une biomasse égale à la biomasse de la strate non pâturable.

3.1.2.1.3 Composition du prélèvement

Elle détermine la composition de l'ingéré, en le répartissant dans chacun des quatre compartiments (Vv, Rv, Vs, Rs), et par différence, la biomasse résiduelle. La composition permet de simuler des comportements de choix de l'animal (sélection pour le vert, pour le végétatif, etc.).

On considère deux manières de gérer la sélection de l'animal :

- Prélèvement proportionnel (pas de sélection) :

  Chaque compartiment est prélevé proportionnellement à ce qu'il représente dans le total. Par exemple, pour une biomasse de 100, composée de 40 Vv, 30 Rv, 20 Vs, 10 Rs, la composition du prélèvement sera 40/30/20/10 %.
  Il est donc nécessaire de recalculer la part de chacun des quatre compartiments dans la biomasse totale afin de calculer la règle de composition de l'ingéré.

- Prélèvement sélectif :

  Les différents compartiments sont prélevés de façon différente de leur représentation dans le total. Ce type de distribution nécessite de vérifier que le prélèvement souhaité est réalisable. Par exemple, pour une biomasse de 100, composée de 40 Vv, 30 Rv, 20 Vs, 10 Rs, avec une intensité de 60 % et une composition de 80/20/0/0, il faudrait prélever 60% de 80, c'est-à-dire 48 Vv alors que la disponibilité n'est que de 40.
  Il apparaît donc nécessaire d'établir des règles de répartition en cas d'impossibilité de satisfaire la demande.

  Il faut tout d'abord vérifier si le prélèvement est réalisable et l'effectuer s'il l'est. Dans le cas contraire, on prélève toute l'offre disponible sauf la proportion équivalente à la strate non pâturable et on répartit la différence selon différentes règles :
  - Soit de façon égale entre les trois compartiments restants
  - Soit de façon préférentielle selon une loi V>R (végétatif prioritaire sur reproducteur) et v>s (vert prioritaire sur sec). La priorité donne un prélèvement deux fois plus important à la catégorie préférée.

Exemple :
Si pour le compartiment Vv, il faut répartir entre Rv, Vs et Rs, l'ingéré non satisfait de 1.2g, les résultats diffèrent selon la règle choisie.
Avec la règle 1, on prélève 0.4g dans chacun des trois compartiments.
Avec la règle 2, V et v prennent 2 points de coefficient et R et s 1 point. Les compartiments Rv, Vs et Rs prennent donc respectivement un coefficient de 3, 3 et 2. La répartition sera donc de 1.2*(3/8)=0.45 g pour Rv et Vs et de 1.2*(2/8)=0.3 g pour Rs.

Il faut à nouveau vérifier si cette quantité de biomasse est disponible dans les différents compartiments et si ce n'est pas le cas, répartir de façon

équivalente le surplus de biomasse entre les compartiments restants, toujours en vérifiant les quantités disponibles.

3.1.2.1.4 Fichier archive

Le fichier archive aura l'extension csv afin de pouvoir l'exporter vers un tableur. Ce fichier comportera les paramètres de la fréquence, puis autant de lignes que de défoliations à réaliser.
Pour chaque ligne, on précisera :

- Biomasse avant défoliation pour dans l'ordre Wcell, WVv, WRv, WVs, WRs
- Option d'intensité choisie H ou W ? (hauteur ou biomasse)
- Intensité de défoliation souhaitée : X% (soit dans le cas de l'option H un équivalent en biomasse de X''%)
- Intensité de défoliation réalisée : X'%
- type de prélèvement = Proportionnel ou Sélectif.
- Composition souhaitée X/Y/Z/K pour respectivement Vv/Rv/Vs/Rs
- règle de report (Règle 1 ou 2)
- Composition réalisée : X'/Y'/Z'/K'

3.1.2.2 Priorité 2 : Description des scénarii

C'est une généralisation de la priorité 1. Elle consiste à modéliser plusieurs périodes de défoliation contrôlée dans un intervalle donné, ce qui revient à créer des scénarii de défoliation. L'interface et le module doivent donc prendre en compte un nombre de périodes de pâturage, ainsi qu'une date de début pour chaque période et les modalités de défoliation qui y sont associées.

Un fichier de sortie devra être généré de la même manière que la priorité 1, c'est-à-dire faire figurer sur chaque ligne le début et la fin de la période de pâturage, ainsi que les variables les plus importantes. Ce fichier aura l'extension csv pour faciliter sa lecture à l'aide d'un tableur.

3.2 Analyse et implémentation du module de défoliation contrôlée

3.2.1 Analyse

Le module de défoliation contrôlée est constitué principalement de trois classes :

- La classe Simulateur est la classe principale. C'est elle qui contrôle le déroulement de la simulation et envoie les ordres de mise à jour et de défoliation.
- La classe Adaptateur permet de communiquer avec le modèle végétal via la classe Parcelle (cf. Figure 15). C'est elle qui va contrôler si le prélèvement est réalisable et établir les règles de répartition si ce n'est pas le cas.
- La classe Paramètres contient toutes les informations nécessaires à la simulation.

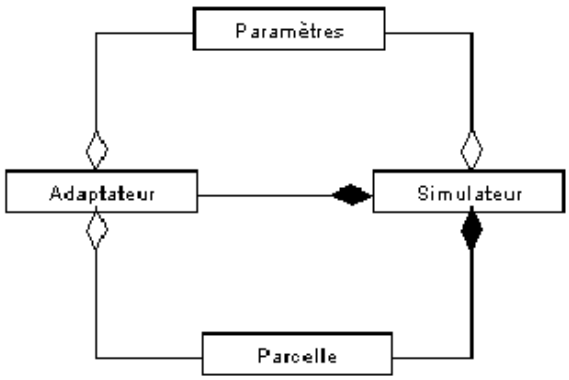

Figure 15 : Diagramme de classes du module de défoliation contrôlée

D'autres classes moins importantes, comme FichierConfigGestion, existent. Celle-ci permet de récupérer les paramètres de la simulation à partir du fichier archive.txt.
La figure suivante montre la relation entre la classe Adaptateur et le modèle végétal.

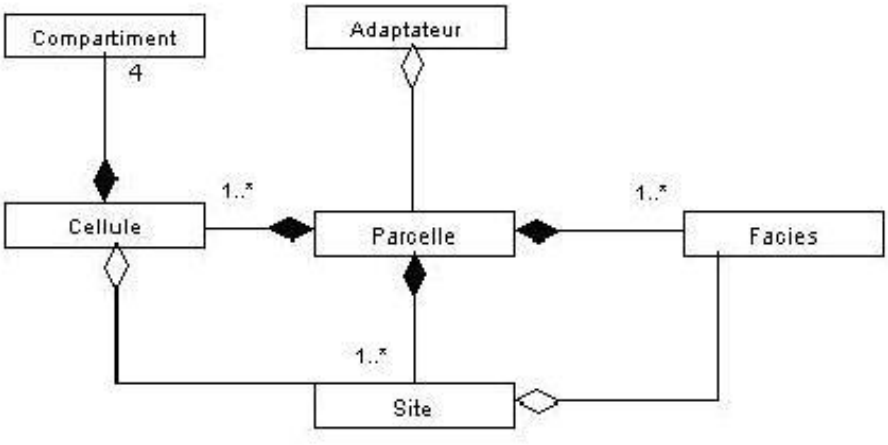

Figure 16 : Diagramme de classes simplifié du modèle végétal avec la classe Adaptateur permettant la communication.

3.2.2 Déroulement de la simulation

La simulation se déroule de la manière suivante :

- Après avoir lu les différents paramètres nécessaires à la simulation, une instance de la classe Simulateur est créée.
- Celle-ci créé ensuite une instance de la classe Parcelle puis une instance de la classe Adaptateur
- Le simulateur appelle la méthode LectureParamètres() puis la méthode initVariables() de la classe Parcelle, afin d'initialiser les différents paramètres du modèle végétal
- Une fois le simulateur créé et initialisé, la simulation est lancée.
- Si le jour courant est un jour de défoliation, la méthode défoliation() de la classe Adaptateur est appelée. Après avoir vérifié si le prélèvement est réalisable et y avoir remédié dans le cas contraire, l'adaptateur appelle la méthode défoliation() de Parcelle.
- La mise à jour du modèle végétal s'effectue en fin de journée par l'appel de la méthode miseAJour() de la classe Adaptateur, qui va elle-même appeler la mise a jour du modèle végétal via la classe Parcelle.
- Une période de défoliation s'arrête lorsque la date de fin ou le nombre de défoliations est atteinte. Si la démarche est simple, la simulation s'arrête. Si c'est une démarche de scénario, la simulation continue jusqu'au nombre de périodes donné, en reprenant le déroulement précédent après le lancement de la simulation.
- A chaque défoliation réalisée, un fichier de sortie sera créé conformément à la description faite dans le cahier des charges.

La figure ci-dessous montre le déroulement de la simulation.

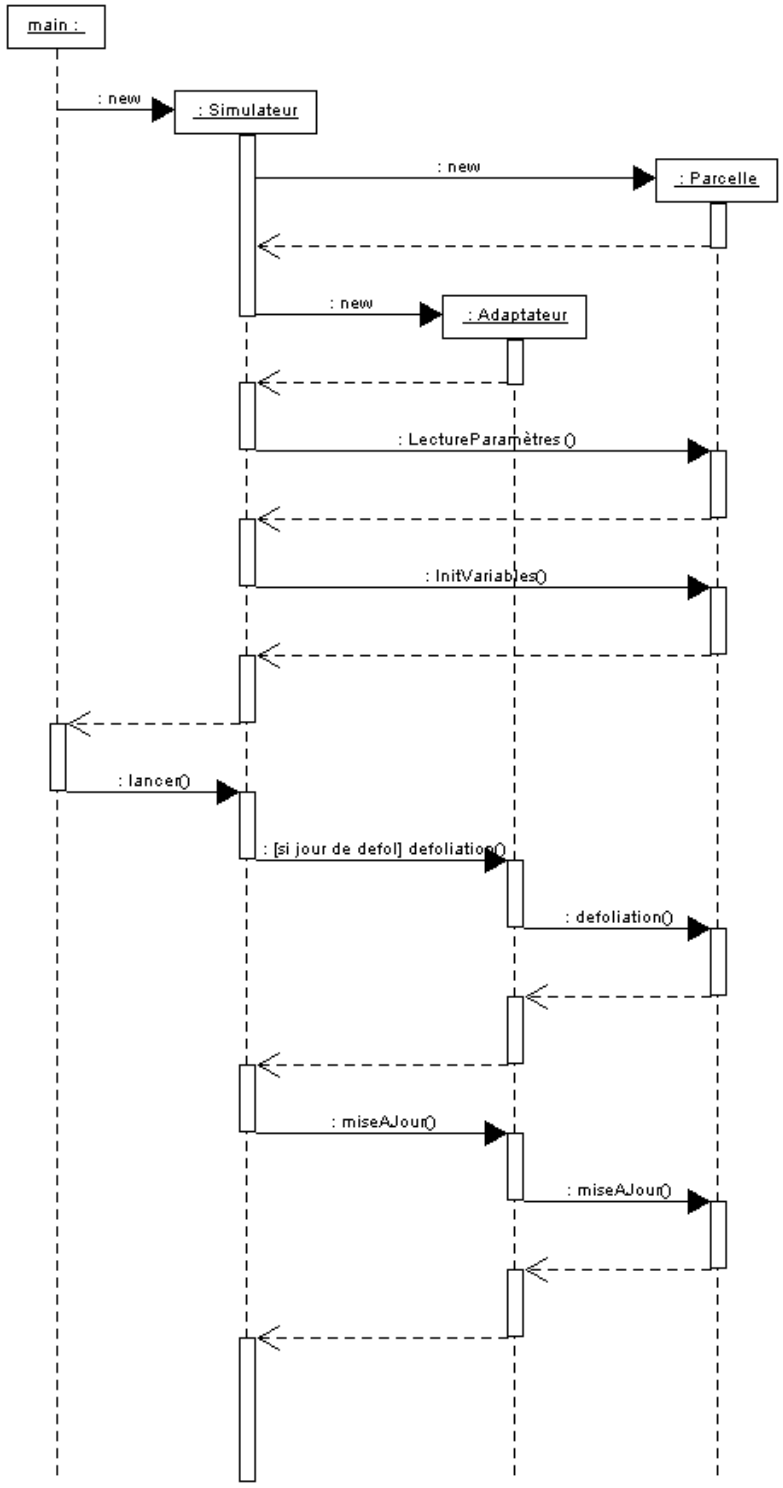

Figure 17 : Diagramme de séquence du module de défoliation contrôlée

3.2.3 Fichiers en entrée

Le module de défoliation contrôlée récupère les paramètres nécessaires à son exécution dans quatre fichiers d'entrée. Ces fichiers ont l'extension csv afin de pouvoir être lus par un tableur.

Le fichier démarche.csv comporte le type de la démarche à effectuer (simple ou scénario) ainsi que le nombre de périodes de défoliations.

```
Démarche;NbPériodes
Simple;1
```

Figure 18 : Exemple de fichier demarche.csv

Le fichier fréquence.csv comporte les paramètres de la fréquence choisie. La première ligne est la gestion choisie, puis chaque ligne à partir de la troisième représente une période de défoliation et les paramètres de la fréquence.

```
Gestion_thermique
Période;DateDépart;Pas;DateFin;NbDefol
Période1;150;70;300;15
```

Figure 19 : Exemple de fichier fréquence.csv

Le fichier intensité.csv contient les informations relatives à l'intensité du prélèvement. Chaque ligne représente une période de défoliation et contient la période, la gestion choisie (hauteur ou biomasse) et l'intensité en pourcentage.

```
Période;Gestion;Intensité
Période1;Biomasse;40
```

Figure 20 : Exemple de fichier intensité.csv

Le fichier composition.csv contient les paramètres de la composition du prélèvement. Chaque ligne représente une période de défoliation et contient cette période et le type de prélèvement. Si celui-ci est de type sélectif, on indique la composition souhaitée respectivement en Vv, Rv, Vs, Rs. S'il est de type proportionnel, la répartition est faite selon la part des compartiments dans la biomasse totale de la cellule, et il n’est pas utile de préciser la composition.

```
Période;Type;VV;RV;VS;RS
Période1;Sélectif;40;30;20;10
```

Figure 21 : Exemple de fichier composition.csv

Le fichier archive.txt, créé par l'interface, est également lu. Il contient les différents paramètres de la simulation comme son nom, sa durée ou encore les chemins et noms des fichiers utiles.

3.3 Interface

3.3.1 Objectif

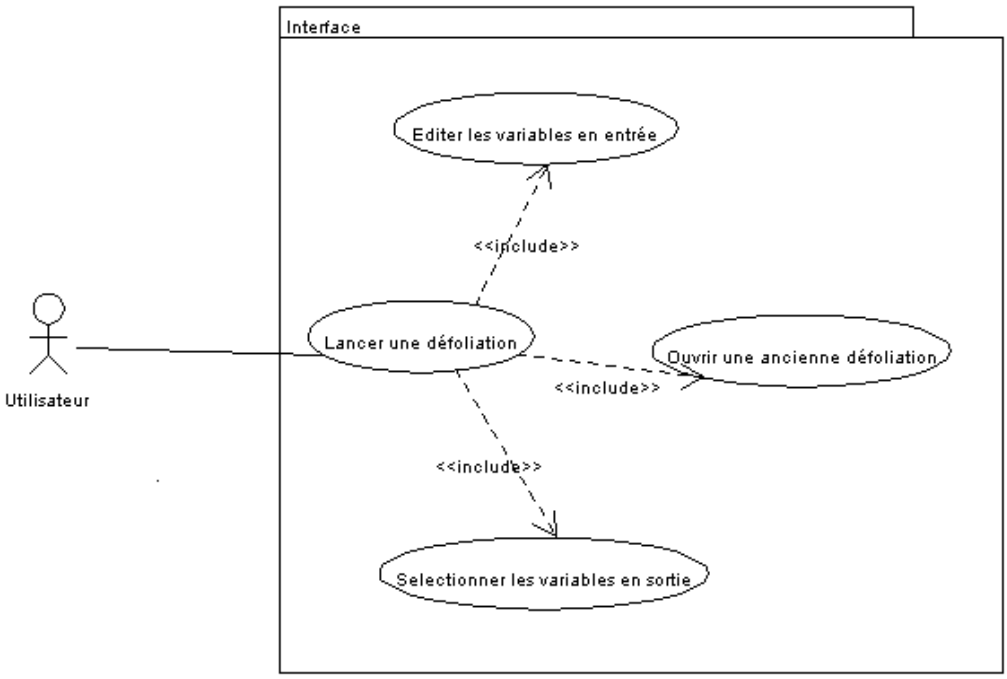

Figure 22 : Diagramme de cas d'utilisation de l'interface

L'interface doit reprendre l'ergonomie de l'interface développée pour le simulateur [PORTAL et SEGUY, 2002]. Nous avons donc choisi de modifier cette interface afin qu'elle convienne au module de défoliation contrôlée. Elle se décomposera donc comme suit :

- Un onglet « Général » contiendra toutes les étapes non spécifiques à la défoliation contrôlée, c'est à dire :

- Une étape « Initialisation » permet de saisir le nom de la défoliation contrôlée ou d'ouvrir une ancienne défoliation via le fichier archive.txt que contient chacune de ces dernières.
- Une étape « Saisie des durées » qui permet de renseigner sur la date de départ de la défoliation contrôlée et sa durée totale.
- Une étape « Fichier faciès » qui permet de sélectionner le fichier faciès qui convient à l'utilisateur.
- Une étape « Fichier site » qui permet de sélectionner le fichier site.
- Une étape « Choix des variables en sorties » qui permet de sélectionner les variables qui apparaîtront dans les fichiers de sorties du modèle végétale.

- Un onglet « Végétal » contiendra toutes les étapes dynamiques relatives aux fichiers d'entrées du modèle Végétale :

- Une étape « FICHIER_CONSTANTE » qui liste les paramètres contenus dans le fichier constant.csv.
- Une étape « FICHIER_ENVIRONNEMENT » qui liste les paramètres contenus dans le fichier environnement.csv.

- Un onglet « Défoliation contrôlée » contiendra toutes les étapes relatives à la saisie des paramètres utiles à la défoliation contrôlée :

- Une étape « Choix de la démarche » qui donne le choix entre la gestion d'une seule période ou d'un scénario. Cette étape est rendue obligatoire par la volonté des utilisateurs de bien séparer ces deux types de gestion de défoliation, relatives aux priorités une et deux.
- Une étape « Fréquence des prélèvements » qui demande tout d'abord le type de gestion (calendaire, thermique, sur les états) puis, en fonction de cela, se renseigne sur chacun des paramètres (date de départ, pas, date finale, nombre de défoliation) et ce pour chacune des périodes.
- Une étape « Intensité des prélèvements » qui donne le choix entre une gestion par rapport à la hauteur moyenne de la cellule ou par la biomasse totale de cette dernière. Après cela, l'utilisateur doit remplir les zones de textes relatives à chacune des périodes renseignant sur le pourcentage de biomasse ou de hauteur à défolier.
- Une étape « Composition des prélèvements » qui permet de choisir tout d'abord entre un prélèvement proportionnel ou un prélèvement sélectif puis ce renseigne, dans ce dernier cas, du pourcentage de VV, RV, VS, RS à prendre en compte, la somme de ces quatre chiffres devant faire bien sûr cent.

3.3.2 Implémentation

Séverine Portal et Romain Séguy ont conçu leur interface de telle manière à ce qu'elle soit facilement réutilisable. Ainsi, chaque onglet correspond à un « sous-système ». Nous avons donc tout d'abord ajouter le sous-système « Défoliation contrôlée » qui contiendra les différentes étapes et vues associées. La gestion des événements et de ces sous systèmes étant elle aussi réutilisable, nous n'avons eu à nous concentrer qu'à l'adaptation au cahier des charges des différentes vues. Ainsi, les quatre vues relatives à la défoliation contrôlée ont été implémentées de façon à ce qu'elles semblent quasi-similaires au point de vue ergonomie. Pour le choix globale de gestion, on utilise une JComboBox qui permet à l'utilisateur de faire un choix dans un menu déroulant. Il apparaît alors dynamiquement autant de ligne que de période à réaliser pour la saisie des données. A chacune de ces vues correspond une étape qui contrôle si les données ne sont pas erronées et écrit les fichiers d'entrées du module de défoliation contrôlée cités plus haut.

Ces quatre étapes rajoutés, il a fallu enlever tout ce qui avait rapport avec l'animal, aussi bien dans les vues que dans les fichiers permettant la génération dynamique de certaines de ces dernières. Ainsi, dans le fichier config_dyn.cvs disparaissent les références à troupeau.cfg et ruminant.cvs. Ces vues ne sont donc plus crées et disparaissent donc de l'interface.

L'appel à la JNI a lui aussi été supprimé car la librairie libcreateur.so, qui permet de générer un fichier .csv à partir d'un fichier .def et d'appliquer alors sur ce fichier nouvellement créé la fonction de perception, n'est plus utile. En effet, aucun animal n'aura à percevoir la cellule et la carte ne comptera qu'un seul site d'une seule cellule ; nous utiliserons donc un unique fichier cellule.csv sans laisser aucun choix à l'utilisateur. En réalité, le choix se fera lors du choix du fichier site qui fera alors correspondre la cellule à un faciès donné. L'utilisateur pourra soit rentrer directement le faciès qu'il souhaite étudier, soit choisir parmi les fichiers .scv contenu dans le dossier Standard.

Enfin, le code a été commenté à l'aide de JavaDoc ce qui permettra l'édition facile d'un manuel pour l'utilisateur désirant modifier l'interface par ses propres moyens.

L'Interface est à ce jour fonctionnelle bien qu'il reste quelque dysfonctionnement (écriture à gauche dans certaine zone de texte, tests plus précis à rajouter lors de la validation de l'étape fréquence de prélèvement).

Partie IV : Résultats et discussion

4.1. Outils utilisés

Pour réaliser ce projet, nous avons utilisé le système d'exploitation Linux, distribution Mandrake 9.1. Le choix de développer sous ce système d'exploitation a été motivé par le fait que le simulateur prairie/troupeau ne soit conçu pour l'instant que pour fonctionner sous Linux, bien que ce dernier puisse être porté sans trop de difficultés sous Windows en modifiant par exemple le nom des répertoires de fichiers d'entrée/sortie.

Pour développer l'interface, nous avons utilisé le freeware Borland JBuilder 7.0 qui avait d'ailleurs été choisi par Séverine Portal et Romain Séguy lors de la conception de leur interface. Il offre un environnement de développement agréable et pratique car il gère de façon dynamique les erreurs, et permet donc des compilations plus sûres. Toutefois, pour l'exécution, nous avons dû passer par la console car JBuilder ne l'autorise pas.

Pour la création des diagrammes UML, il a été utilisé le logiciel Poséidon for UML Community Edition 1.51. Ce langage de modélisation est compris par la plupart des chercheurs, utilisé depuis longtemps pour ce simulateur et considéré comme la norme dans le domaine objet.

4.2. Démarche de travail

La première étape de notre travail a consisté à comprendre le simulateur et ses enjeux pour les biologistes. Ainsi les premières réunions nous ont permis d'acquérir le vocabulaire et les principes indispensables à la compréhension du modèle.

Nous nous sommes donc tout d'abord attaqués, pour l'un aux modifications des fonctions de montaison et de sénescence, pour l'autre à l'apprentissage du langage Java afin de créer l'interface graphique.

Puis la conception du module de défoliation contrôlée a véritablement eu lieu en procédant par suppression et ajout de code successif, ceci dans un but de prudence.

4.3. Difficultés rencontrées

La première difficulté rencontrée a été provoquée par le fait que ce simulateur a été développé depuis plusieurs années et n'a aucun manuel global de fonctionnement. Il nous a fallu lire beaucoup de rapports dont de nombreuses informations se contredisent, de par le fait que le modèle a subi de nombreuses modifications dans le temps. Ainsi, nous avons compris seulement à la fin pourquoi l'interface n'appelait plus le simulateur via la JNI, ce qui pourtant était affirmé dans le rapport de l'interface [PORTAL et SEGUY, 2003]. En fait, le problème venait du fait que, *via* l'appel à une librairie, le simulateur ne pouvait gérer une simulation de plus de vingt jours à cause d'un problème de taille de pile. L'interface avait donc été modifiée et fait maintenant appel à un exécutable. Un fichier archive contient donc les paramètres utiles à la simulation et le simulateur les lit via une nouvelle classe FichierConfigGestion.

De plus, le code C++ de ModVege manque de commentaires et d'homogénéité, mis à part les parties les plus récentes. D'ailleurs, Séverine PORTAL avait commencé un travail de « dépoussiérage » du code et a fixé un guide de style (Codage professionnel en C++ : proposition de style de codage).

En ce qui concerne les difficultés relatives au domaine de la biologie, nous avons tout d'abord dû nous familiariser avec le vocabulaire ainsi que le fonctionnement des différentes fonctions biologiques que nous avons modifiées.

4.4. Futur du module de défoliation

Les fonctions de montaison, de sénescence et de litière ont été implémentées et sont en attente d'être validées. Nous les avons testées et les données récupérées restent dans l'ordre de grandeur des valeurs précédentes.

En ce qui concerne le module de défoliation et son interface, la priorité une a été implémentée et la deuxième est sur le point d'aboutir mais certains problèmes pourraient se poser. En effet, selon le type de fréquence choisie, il n'y a pas correspondance entre les unités de la date de départ de la période et la date de fin ce qui peut créer des chevauchements. Par exemple, la gestion thermique prend comme date de départ une somme de température depuis le début de l'année alors que la date de fin prend un nombre de jours, il n'y a donc pas de relation directe entre ces deux paramètres ce qui empêche toute vérification lors de la saisie dans l'interface. Nous devrons donc faire un choix et homogénéiser les unités des différents paramètres.

Ce module pourra être utilisé comme outil de validation du modèle végétal et permettra de mieux considérer l'effet propre du prélèvement par l'animal sur la végétation ainsi que l'évolution de ses états et de ses caractères fonctionnels.

Conclusion

Les objectifs demandés au début du projet sont presque totalement atteints. En effet, les deux fonctions modifiées sont effectives mais n'ont pas encore été validées. Le module de défoliation contrôlée, aidée de son interface dédiée, fonctionne dans le cadre de la priorité une et est sur le point d'intégrer la priorité deux.

En plus des bénéfices retirer dans le domaine informatique, ce projet nous a beaucoup apporté au niveau des méthodes de travail. En effet, nous avons appris à collecter des informations dans une masse de documentation importante. De plus, nous avons pu mettre l'outil informatique au service d'un autre domaine scientifique : la biologie, qui était jusqu'alors inconnue pour nous.

Enfin, nous avons appris à nous intégrer à un projet de grande ampleur, ce qui nous a permis de réaliser un travail intéressant et motivant, mais ce qui nous a aussi obligé de relire du code « étranger » et à positionner notre façon d'implémenter par rapport à ce dernier.

Pour conclure, ce module prend place dans une optique non-perenne par rapport au projet. Il donne au chercheur un outil efficace pour analyser le comportement de la végétation modéliser par ModVege suite à des défoliations. On se place donc ici du point du végétal, et non plus de celui de l'animal.

Références bibliographiques

[BAUMONT et coll. 2002] BAUMONT R., DUMONT B., CARRERE P., PEROCHON L., MAZEL C., FORCE C., PRACHE S., LOAULT F., SOUSSANA J.F., HILL D., PETIT M., 2002. Développement d'un modèle multi-agents spatialisé d'un troupeau de ruminants pâturant une prairie hétérogène. *Rencontres Recherches Ruminants, 9, 69-72.*

[BOICHAT 2003] BOICHAT J-B., Apprendre JAVA et C++ en parallèle, troisième édition, édition EYROLLES

[CARRERE et al. 2002] CARRERE P., FORCE C., SOUSSANA J.F. , LOUAULT F. , DUMONT B. and BAUMONT R. (2002). Design of a spatial model of a perennial grassland grazed by a herd of ruminants : the vegetation sub-model. EGF 2002. Grassland Science in Europe : 7, 282-283.

[GAMMA et al.] GAMMA E., HELM R., JOHNSON R., VLISSIDES J., Design Patterns, Element of Reusable Object-Oriented Software, Software Development, pages 315-323

[HILL 2003] HILL D.R.C., Fondements du langage JAVA (et comparaison avec C et C++), Cours de Java à l'ISIMA, 2003

[MARTIN 2002] MARTIN G., Analyse et conception du couplage entre les modules animal et végétal du simulateur troupeau/parcelle, rapport de stage INRA 2002

[PEROCHON et al. 2001] PEROCHON L., CARRERE P., BAUMONT R., DUMONT B., MAZEL C., FORCE C., HILL D., D'HOUR P., LOUAULT F., PRACHE S., SOUSSANA J.F., PETIT M. (2001) Design of a spatial multi-agent model of a perennial grassland ecosystem grazed by a herd of ruminants**.** Simulation in industry'2001, 13th European Simulation Symposium 2001, N. Giambiasi and C. Frydman (Eds.). October 18-20, 2001, Marseille, France. pp 509-513

[PORTAL et SEGUY 2003] PORTAL S., SEGUY R., Interface graphique pour un simulateur parcelle/troupeau, rapport de projet 2003

[PORTAL 2003] PORTAL S., Conception du module spatial et social d'un simulateur parcelle/troupeau, rapport de stage INRA 2003

[SAUVANT 1996 et al.] SAUVANT D., BAUMONT R., FAVERDIN P., Development of a Mechanistic Model of Intake and Chewing Activities of Sheep, Journal of Animal Science, 74, 1996, pages 2785-2802