Unité BIA, INRA Toulouse

Université du Littoral
Côte d'Opale à Calais

Modélisation et Simulation de la décision

Pilotage d'un système de production agricole

29 août 2008

Encadrant
Mr Éric Ramat

Tuteurs
Mr Frédéric Garcia, Mr Roger-Martin Clouaire, Mr Gauthier Quesnel

Étudiant
Weisenhorn Geoffroy

Remerciements

Je tiens tout d'abord à remercier Frédéric Garcia, Roger-Martin Clouaire, Gauthier Quesnel et Eric Ramat pour m'avoir proposé ce sujet de stage et m'avoir accepté en tant que stagiaire. Je les remercie également pour m'avoir guidé, motivé tout au long de ce stage.

Merci également à Ronan Trépos qui m'a apporté son aide pour la compréhension du projet SACADEAU.

Je voudrais remercier toute l'équipe de BIA ainsi que mes collègues de bureau pour y avoir fait régner une ambiance conviviale, détendue mais néanmoins travailleuse.

Table des matières

CHAPITRE 1

Introduction

Les objectifs du stage porte sur l'étude de la modélisation des processus de décision en environnement dynamique et incertain dans le cadre de modélisation de *DEVS* de la simulation à événements discrets.

La discipline de l'informatique qui s'occupe de la simulation de la décision est l'intelligence artificielle. Différentes approches existent dans la façon dont le système décisionnel réalise ce processus :
- décision de type réactif à base de règles de décision ;
- décision de type délibératif a base de plans d'actions et d'ordonnancement ;
- décision de type mixte avec des plans réactifs ;
- décision basé sur une politique markovienne.

La modélisation de système décisionnel varie selon la classe de problèmes (décision séquentielle dans l'incertain, amélioration ou apprentissage de politique,. . .), l'incertitude du système piloté et le temps.

Nous présenterons dans une premier partie, le contexte de ce travail c'est-à-dire les concepts de la modélisation et de la simulation puis les problèmes liés à la conduite d'un système de production agricole. Dans une deuxième partie, nous verrons une synthèse des formalismes de la décision, à travers des articles relatifs au domaine. Pour la troisième partie, nous expliquerons le modèle décisionnel que nous avons décidé de reformaliser dans le cadre DEVS. Pour finir par une description d'un langage de spécification de plan de production accompagné d'une explication sur son implémentation. Nous conclurons ce travail par des pistes pour améliorer le modèle développé.

CHAPITRE 2

Présentation de l'INRA

2.1 Présentation générale

L'INRA (Institut Nationale de la Recherche en Agronomie) est un organisme public créé en 1946, avec pour mission de *mettre la science et la technologie au service du développement de l'agriculture* en améliorant les techniques de production (culture et élevage) et la sélection génétique végétale et animale.

En 2007, L'institut a maintenu son 2^{e} rang mondial en termes de publications scientifiques dans le domaine de la recherche agronomique. Aujourd'hui, ses recherches concernent trois domaines fortement imbriqués : l'alimentation, l'agriculture et l'environnement avec l'ambition de développer une agriculture à la fois compétitive, respectueuse de l'environnement, des territoires et des ressources naturelles. Les orientations scientifiques de l'INRA sont dessinées par les questions scientifiques et leurs interfaces avec les demandes et les évolutions de la société relatives à ses domaines de recherches. Les questions de recherche actuelles nécessitent le plus souvent des approches pluridisciplinaires, adaptées à l'étude d'un système global de production et d'échanges mondiaux, avec la mise en commun de forces de recherche dans le cadre de partenariats.

Les orientations scientifiques 2006-2009 s'organisent autour de 6 axes :
– Un environnement préservé, un espace rural vivant ;
– Une alimentation saine et équilibrée ;
– Des produits transformes compétitifs et de qualité ;
– Une connaissance approfondie du vivant ;
– Des systèmes de production innovants et durables ;
– L'analyse des filières et des politiques publiques.
Le centre INRA Toulouse Midi-Pyrénées est l'une des 21 implantations de l'INRA en France, impliqué dans un grand ensemble universitaire et scien-

tifique (8500 chercheurs, 100 000 étudiants, 4 universités et 13 Écoles d'ingénieurs). L'INRA s'organise en **Unité**.
Le site de Toulouse s'architecture autour des trois grands domaines de recherche : le génome et les biotechnologies, la sécurité des aliments, le territoire et les produits.

2.2 Présentation de l'unité BIA

Le site de Toulouse héberge entre-autre le département *Mathématiques et Informatiques Appliqués* (MIA) qui vise à développer et à promouvoir l'utilisation de méthodes originales de statistique, analyse de systèmes et d'intelligence artificielle. Ce département comporte 8 unités de recherche dont l'unité *Biométrie et Intelligence Artificielle* (BIA) dans laquelle le stage s'est déroulé.

Cette unité est composée de deux équipes de recherches qui se focalisent sur les deux thèmes suivants :
- l'analyse et la gestion des systèmes agricoles, forestiers et naturels
- l'analyse des organismes de bactéries, plantes et animaux par des approches génomiques et génétiques

Ces recherches s'accompagnent d'une activité de production de logiciels pour leur valorisation et d'une activité de formation pour leur diffusion.
Plus particulièrement, les thèmes de recherche se concentrent sur :
- l'apprentissage et l'optimisation de stratégies mises en oeuvre par les acteurs des agro-écosystèmes ;
- la construction de modèle de décision dans les agrosystèmes et leur simulation par couplage avec des modèles biophysiques ;
- la modélisation statistique de dynamiques spatio-temporelles pour la désagrégation d'informations satellite ;
- l'analyse statistique de processus spatiaux dans les agro-ecosystèmes à à partir d'informations à plusieurs échelles ;
- la construction de cartes génétiques et de cartes d'hybrides irradiés ;
- la détection de zones du génome associées à un caractère quantitatif ;
- la recherche sur la séquence de gènes de protéines et d'ARN.

2.3 Présentation du projet RECORD

Le projet *RECORD* (REnovation et COoRDination de la modélisation de cultures pour la gestion des agrosystèmes) (http://record.toulouse.inra.fr) est une plate-forme informatique de modélisation et simulation pour l'étude des systèmes de culture. La plate-forme a été initiée en 2005 par les départements *Environnement et Agronomie* (EA) et *MIA* pour répondre à un objectif central :

> « Mettre au point des systèmes de cultures innovants capables d'assurer des fonctions agronomiques et environnementales spécifiées ».

La modélisation et la simulation sont des voies privilégiées pour la conception mais également l'évaluation des systèmes de culture. Le but de la plate-forme *RECORD* est de dépasser les limites des approches actuelles de modélisation du point de vue logiciel sur plusieurs critères : la complexité des modèles à implémenter, les difficultés à échanger, partager ou coupler des modèles et mutualiser des besoins communs en outils de simulation. C'est un outil à destination des scientifiques agronomes travaillant sur l'analyse et la conception de systèmes de cultures : concepteurs de modèles, développeurs et utilisateurs.
La solution technique, décrite dans [Chabier *et al.*, 2007], adoptée pour le projet *RECORD*, est une plate-forme informatique de multi-modélisation et de simulation de systèmes complexes : *VLE* [Quesnel *et al.*, 2007]. La plate-forme dispose d'un ensemble d'outils et de bibliothèques permettant le couplage et la simulation des modèles hétérogènes, ce qui signifie que les modèles sont spécifiés dans des formalismes différents.

VLE repose sur les bases théoriques et opérationnelles de la modélisation et de la simulation définies par **B. P. Zeigler** [Zeigler *et al.*, 2000] dans *DEVS*, un cadre formel de modélisation et de simulation à événements discrets, décrit plus en détails dans la suite du rapport. *VLE* est développé par plusieurs laboratoires : BIA, LIL (Laboratoire d'Informatique du Littoral) et le CIRAD[1] (**R. Duboz et J.C. Soulié**), distribué sous la licence libre *GNU GPLv3*. L'ensemble de la plate-forme a été développé en C++ en utilisant des bibliothèques standards et multi-plateformes : *STL* et *Boost*. Une interface graphique est disponible, indépendante du noyau de simulation, utilisant des bibliothèques C++ de *GTK+* : *gtkmm* et *glademm*.

En résumé, les caractéristiques clés de la plate-forme *RECORD* sont :
- Une plate-forme pour construire et travailler avec des modèles ;
- Une bibliothèque de modèles sera fournie ;
- La définition d'un cadre formel de modélisation commun à tous les modèles : c'est un couplage au niveau des modèles qui est réalisé et non au niveau logiciel ;
- L'implémentation des modèles sera spécifiée dans le langage de programmation C++ ou Python ;
- C'est aussi une plate-forme pour l'analyse et la conception par simulation (estimation, multi-simulation, analyse de données, visualisation et optimisation).

[1] CIRAD : Centre de coopération internationale en recherche agronomique

CHAPITRE 3

Cadre de l'étude

Ce travail a été réalisé au cours d'un stage de Master 2 Ingénierie du Logiciel Libre. L'intitulé du stage est « *l'étude de la modélisation des processus de décision en environnement dynamique et incertain dans le cadre théorique DEVS de la simulation à événements discrets* ».

3.1 Contexte de travail

Ce travail a permis de réunir deux domaines, à savoir, d'une part, la modélisation et la simulation dans le formalisme *DEVS* et d'autre part, les processus de décision.

L'étude des agro-écosystèmes nécessite un investissement sur les problèmes de décisions. Il est extrêmement important de comprendre les caractéristiques de ces systèmes, auxquels les chercheurs en agronomie sont confrontés.

L'un des objectifs de la plate-forme *RECORD* est de pouvoir modéliser et simuler le comportement décisionnel d'acteurs opérant sur des agro-écosystèmes. *VLE* dispose aujourd'hui de plusieurs extensions *DEVS* permettant la représentation et le couplage de formalismes qui conviennent au besoin de la modélisation d'agro-écosystèmes, tel les équations aux différences, les équations différentielles ordinaires et spatialisées, les automates cellulaires et les processus semi-Markoviens généralisés. Toutefois il serait également nécessaire d'intégrer des formalismes permettant de représenter des processus de décision, selon différentes approches : des approches de types réactifs à base de règles de décision, de politiques Markoviennes, délibératif à base de plans d'actions et d'ordonnancement, ou bien mixte avec des plans réactifs.
Dans cette partie, nous définissons tout d'abord les différents concepts employés issues de la modélisation et de la simulation. Une deuxième partie

décrira plus amplement le contexte des systèmes de production agricole ainsi que les concepts associés à la conduite de culture.

3.1.1 Concepts relatifs à la modélisation et la simulation

La modélisation et la simulation sont des outils pluri-disciplinaires pour l'étude de systèmes dynamiques, par exemple un agro-écosystème. Un système dynamique est un système évoluant selon une variable temporelle. Les caractéristiques d'un tel système sont : un état courant et une fonction qui décrit le changement d'état par rapport au temps. La modélisation d'un système consiste à créer un objet abstrait, le *Modèle*, représentant la structure et la dynamique mimant au mieux les propriétés du système à modèliser. La conception de modèle s'appuie sur ce qui est appelé un formalisme. Selon *Ludwig Klages*, « *le formalisme, est la pensée par signes purs* ». Les programmes informatique en sont de beaux exemples. Toujours selon Klages,[1]

> « *Le but de la pensée formaliste, c'est : des résultats de la pensée atteints sans l'effort de la pensée, des réponses trouvées sans l'intermédiaire de la recherche, la domination de l'Esprit établie sans le moyen et l'instrument de la conscience, qui dépend toujours pour une part de la Vie. . . .*[2] ».

Dans le domaine de la modélisation, le formalisme est l'outil permettant de spécifier la fonction de changement d'états du système modélisé. De façon générale, un formalisme est l'outil opérationnel du paradigme. Le paradigme est un concept utilisé au niveau sémantique pour spécifier un modèle, auquel est associé un ou plusieurs formalismes. Dans la suite du rapport, divers exemples de formalismes adaptés pour la modélisation de systèmes décisionnels seront présentées.
Une simulation est une expérience réalisé sur un modèle à condition qu'il soit d'une part contrôlable (les entrées) et observable (sorties). Il est important de noter que la description d'une expérience et la description d'un modèle sont des entités conceptuellement différentes. Les informations issues du résultat d'une simulation sont fortement dépendant de la manière dont le système a été modélisé. Les motivations pour utiliser une simulation sont multiples :

- les expériences sont trop coûteuses, trop dangereuses ou bien le système n'existe pas encore,
- l'échelle de temps de la dynamique du système n'est pas adéquate pour un expérimentateur,
- certaines variables ne sont pas forcément accessible dans un système réel. En simulation toutes les variables peuvent être contrôlées,
- la possibilité d'isoler certains effets en particulier les bruits ou des effets de bord secondaires

[1] Ludwig Klages, les principes de la caractérologie, 1950

[2] . . . Sans doute, le parfait formaliste serait un appareil de précision sans conscience, capable d'une variété de réactions inquiétante et qu'on pourrait alors composer, soit dans un atelier de construction, soit dans un alambic, comme un homonculus

La famille des modèles mathématiques se divise en plusieurs catégories selon leurs caractéristiques. Une caractéristique importante est la dépendance avec le temps : *dynamique* ou *statique*. Un modèle peut évoluer de façon *continue* dans le temps ou bien à des points *discrets*. Certains phénomènes naturels sont décrits par des processus stochastiques ou par des lois probabilistes. Ce type de modèles sont qualifiés de *stochastique* [3] ou *probabiliste* car c'est à l'aide d'outils statistiques qu'il est possible de représenter correctement la réalité de leurs comportements. C'est une discipline faisant souvent appel au raisonnent inductif : à partir d'un certain nombre d'observations élémentaires, une loi générale est construite pour *expliquer* les observations. Cependant en raison du caractère inductif, les résultats obtenus par la statistique peuvent être remis en question par de nouvelles observations. Les résultats obtenus seront justifiés dans la mesure où ils sont opérationnels et non pas pour leur représentation de la vérité absolue. Par opposition, les modèles *déterministes* peuvent être décrient sans incertain[4].

3.1.2 Conduite d'un système de production agricole

La conduite d'une production agricole est une tâche notoirement complexe (voir [Martin-Clouaire and Rellier, 2003] et [Chatelin *et al.*, 2007]) dans le sens où le comportement du système piloté dépend beaucoup d'éléments exogènes, la météo étant le plus important d'entre eux. Pour l'agriculteur, conduire un système de production, consiste à choisir et mettre en œuvre au cours du processus de production les décisions successives d'opérations techniques à réaliser en fonction de la situation rencontrée, des objectifs et des différentes contraintes.

Deux concepts agronomiques doivent être définis pour une bonne compréhension du sujet : le concept *Système de culture* et le concept d'*Itinéraire technique*.

Le concept agronomique de *système de culture* [Sebliotte, 1990] :

Définition. Un système de culture est l'ensemble des modalités techniques mises en œuvre sur des parcelles traitées de manière identique. Chaque système de culture se définit par :
– la nature des cultures et leur ordre de succession ;
– les itinéraires techniques appliqués à ces différentes cultures, ce qui inclut le choix des variétés pour les cultures retenues.

Cette définition fait intervenir un autre concept agronomique qu'il est également important de comprendre, pour la suite, celui d'*Itinéraire technique* [Sebliotte, 1974] :

Définition. Un itinéraire technique est une combinaison logique et ordonnée de techniques culturales utilisées sur une parcelle, qui permet, par le

[3] Introduction aux processus stochastiques http://www.montefiore.ulg.ac.be/~lwh/ProcStoch/

[4] Un phénomène stochastique peut être également modélisé de façon déterministe

contrôle du milieu écologique, d'atteindre un objectif de production donné, en quantité et en qualité.

Les outils de modélisation et de simulation sont des outils fondamentaux pour étudier plus facilement le fonctionnement des systèmes de culture, pour expérimenter des manières innovantes une manière de conduire de tels systèmes pour satisfaire des contraintes économiques, sociales ou écologiques. Le pilote du système de production repose sur la mise en œuvre d'une *stratégie de conduite*. Cette stratégie spécifie comment doivent être planifiées les différents actes techniques décrient dans l'itinéraire technique (voir figure 3.1), et comment cette organisation doit être adaptée dans des situations identifiées.

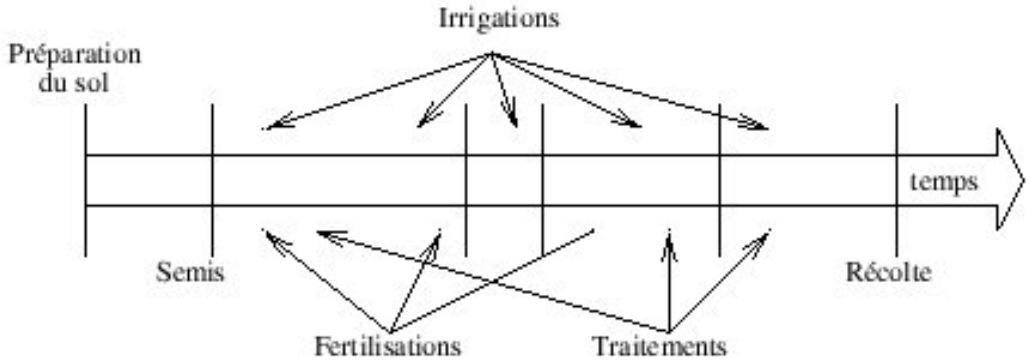

FIG. 3.1: *L'itinéraire technique dans une saison culturale [Crespo, 2008]*

Afin d'étudier le fonctionnement d'un système de production, le processus de décision et ses interactions avec le système biophysique doivent être également modélisé. La figure 3.2 schématise une point de vue systémique d'un système de production représentant une production de tomates sous serre. Le système est décomposé en trois sous-systèmes : le pilote, le système opérant et le système biophysique. Les informations sur l'état du système sont partiellement accessibles par les autres représentées par des flèches en pointillées. La modélisation du processus de décision implique la possibilité de représenter le déroulement des actions résultant de la prise de décision, et les facteurs importants de l'environnement extérieur.

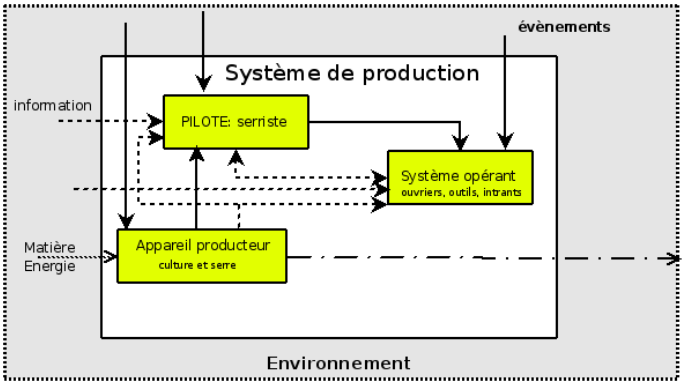

FIG. 3.2: *Système agricole [Martin-Clouaire and Rellier, 2006]*

Les décisions de l'agriculteur sont prises en fonction de la stratégie de conduite et de l'état perçu ou prévisible de l'appareil biophysique, des ressources disponibles pour la mise en œuvre d'une opération culturale et de l'occurrence d'évènements externes incontrôlables.
La nature biologique des productions agricoles les rend fondamentalement différentes des productions manufacturières. L'incertitude dans l'industrie concerne les objectifs de production (la demande) et dans une moindre mesure la disponibilité des ressources. Dans l'agriculture, l'incertitude concerne les facteurs exogènes, tel que le climat ou le marché forçant l'enchaînement des actions à être dépendant du contexte pour saisir une opportunité ou prévenir une menace. Les processus de production en industrie ont été conçus par l'homme et par conséquent les problèmes d'ordonnancements des opérations ont déjà été pris en compte durant la conception de la chaîne de production. En agriculture, les processus de production doivent se superposer à des processus biologiques qui ne sont que partiellement contrôlables.

3.2 Problématique

La problématique de ce travail est de savoir si un modèle décisionnel existant, parmi les modéles de simulation de conduite de culture, peut être modèlisé en *DEVS* en utilisant des formalismes de décision existants.

Cette étude s'est faite en deux parties. Dans un premier temps, il s'agit de comprendre différents formalismes de modélisation de la décision à travers diverses publications. Ensuite nous avons étudié la représentation d'un modèle (SACADEAU) décisionnel dans le cadre de modélisation *DEVS*.

CHAPITRE 4

État de l'art sur les formalismes de la décision

Cette partie est destinée à faire un état de l'art du domaine de la décision à travers diverses publications traitant de ce thème. Nous allons restreindre le domaine en nous intéressant aux formalismes pouvant être utilisables dans la modélisation et la simulation d'un système décisionnel.

4.1 Problèmes de la prise de décision

4.1.1 La prise de décision

La prise de décision est un processus cognitif complexe visant à la sélection d'un type d'action parmi différentes alternatives. C'est une méthode de raisonnement pouvant s'appuyer sur des arguments rationnels mais également irrationnels. Le processus se déclenche par un individu (ou plusieurs) ressentant le besoin d'agir sans savoir comment diriger l'action.
La citation suivante de [Kepner and Tregoe, 1997] décrit le problème de la prise de la décision :

> Le processus décisionnel est difficile parce qu'il nécessite non seulement de l'expérience, de la connaissance, un sens commun et un jugement, mais implique beaucoup d'incertitudes découlant de l'action qui sera décidée.

Nous pouvons distinguer trois grandes approches du concept de décision :
- La première approche est de considérer la décision comme un choix de type optimisateur ne prenant pas en compte ni le décideur et ni le contexte. Cette branche est plus communément appelée *Recherche Opérationnelle*
- Une autre approche est de prendre en compte la dimension cognitive des décideurs, en particulier les limites de rationalité, on parle de *processus de décision*. Il s'agira toujours d'un choix entre solutions potentielles, qui se fondera sur un critère de satisfaction. Les neurosciences ont mit en lumière *les confrontations entre la cognition et l'émotion dans le processus*

de décision et le rôle des zones cérébrales correspondant à la souffrance et au plaisir. Il est ainsi difficile d'étudier le processus de décision seulement d'un point de vue rationnel.
– La dernière approche plus récente, intègre, outre le décideur, le contexte dans lequel se déroule la décision, on parle de *Naturalistic Decision Making*. La démarche est inverse, le comportement du décideur est modélisé suite à l'observation de celui-ci en situation. Dans ce cas, la décision n'est plus un choix entre plusieurs alternatives mais la capacité d'un décideur à reconnaître la situation dans laquelle il se trouve, dépendant fortement de son expérience.

4.1.2 Le processus de décision : Raisonner pour décider

L'origine de toutes décisions provient d'une insatisfaction, que l'on appelle *Problème de décision*. Cette insatisfaction provient de la différence entre l'état actuelle du *monde*[1] et un autre état désiré qui n'existe pas.

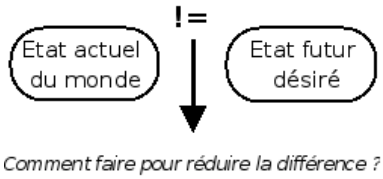

FIG. 4.1: *Problème de décision [Pomerol, 1995]*

Dans un contexte où les choses sont connues avec certitude, l'état *actuelle* est perçu comme unique pour une personne[2]. De plus, l'état désiré peut être très difficile à atteindre compte tenu de l'antagonisme entre les propriétés attendues.

Avant la phase de décision, une personne a sa propre perception de l'état actuel mais tentera de l'identifier par référence à son expérience. La première phase d'une décision est de trouver une ou plusieurs situations similaires à la perception de l'état présent dans le passé. Cette phase se nomme *Pattern matching* ou *Diagnostic* selon le contexte ou la complexité de l'état étudié.

Cependant dans certains cas, il est très difficile de savoir ce qui s'est passé et la situation exacte, l'utilisation d'outils de probabilité permet d'avoir un diagnostic incertain. Il est important de ne pas oublier de considérer le futur dans un prise de décision, c'est-à-dire : quel est la situation vers laquelle on souhaite se diriger ?
Le diagramme suivant 4.2 décrit les principaux ingrédients d'un raisonnement décisionnel : la phase de *diagnostique* et la phase d'anticipation.

[1] le terme *monde* représente tout ce qui est extérieur à l'état mental d'un agent décideur (point de vue subjectif)
[2] Il est intéressant de noter la dimension personnelle qui apparaît à l'origine du problème, car ce que désire une personne n'est pas désiré par une autre.

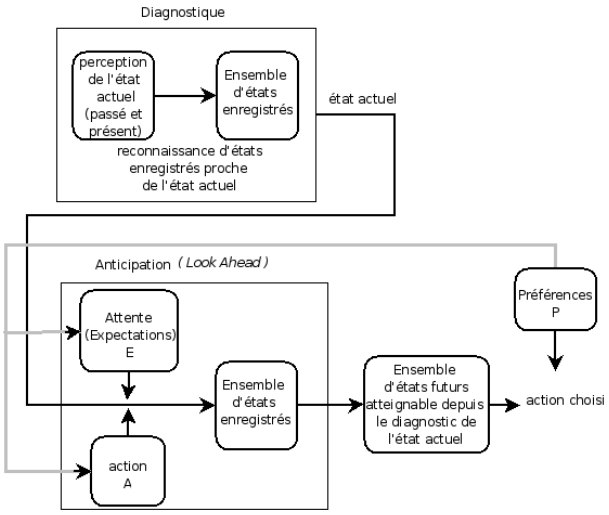

FIG. 4.2: *Processus de décision [Pomerol, 1995]*

Cette séparation n'est pas toujours facilement identifiable dans l'implémentation du système décisionnel. Ce diagramme souligne le fait qu'un ensemble d'actions ne peut être changé durant le processus de décision. Dans certains cas, le fait de relier les préférences avec les attentes du système, est discutable car cela signifie que le futur est vu selon des préférences qui doivent vraisemblablement être évitées dans un processus de décision rationnel.

4.2 Approche de la décision en IA

Le problème de la prise de décision est un domaine où l'intelligence artificielle peut apporter des réponses. Le sujet de la modélisation de processus de décision s'intéresse à modéliser le raisonnement qui conduit à une décision. Plusieurs approches ont été développées dans le domaine de l'IA afin de simuler un raisonnement sur des approches symbolique, connexionniste et statistique. Quelques approches de modélisation de la décision sont présentées dans cette partie. Elles sont adaptées à des problèmes précis, chacun proposant des améliorations dans la manière de modéliser un système cognitif.
L'objectif de l'IA est de permettre aux machines d'être capables de réaliser des tâches considérées comme « intelligentes ». Nous pouvons distinguer trois types de tâches élémentaires considérées comme « intelligentes » : la prédiction, la planification et l'explication [Castilho, 1998].

- La *prédiction* est la tâche la plus souvent présente dans les travaux du domaine des raisonnements sur les actions. Elle consiste à trouver l'état du monde après l'exécution d'une séquence spécifique d'actions.
- L'*explication* est le chemin contraire, trouver l'état du monde avant l'exécution d'une séquence d'actions.
- La *planification* est la tâche qui cherche à savoir s'il existe une séquence d'actions qui peut nous amener d'une représentation du monde actuel

à une représentation d'un monde désiré : le *But*. Si c'est possible, nous voudrions connaître la séquence d'actions trouvée pour arriver au but. Cette séquence d'actions s'appelle un *Plan*.

La formalisation du raisonnement dans un processus de décision s'articule autour de la notion d'*action* [Castilho, 1998]. Cependant il est nécessaire de s'intéresser de près au sujet de la *représentation* du savoir sur le monde. En effet, les systèmes qui raisonnent sur les problèmes réels dans un monde dynamique ne peuvent représenter qu'une partie de la réalité. Chaque représentation n'est qu'une simplification des objets et des relations qui peuvent être pertinents dans un problème de prise de décision. L'avantage des modèles IA est de pouvoir calibrer les manques d'informations. L'IA est particulièrement adaptée à une perspective *Weberienne*, c'est à dire qui définit le décideur comme quelqu'un qui agit en pensant et interprétant son environnement [Schneider, 1996].

4.2.1 Principes de la théorie de la décision

La théorie de la décision s'appuie sur les axiomes de probabilité et d'utilité. Les probabilités sont un outil puissant permettant d'expliciter les notions de croyance partielle sur des informations incomplètes. La théorie de la décision étend l'outil de la probabilité en évaluant chaque alternative. Si une proposition a une probabilité de 1, il est pertinent de croire en cette alternative. La théorie de la décision fournis les principes pour l'inférence rationnelle et la prise de la décision dans l'incertain.

Quant à la théorie de l'utilité, elle apporte pour sa part, un ensemble de principes pour rendre cohérentes les préférences et les décisions. Les préférences sont les résultats de l'évaluation de l'agent pour des états du monde. La théorie est fondée sur un ensemble d'axiomes simples et de règles concernant le choix dans l'incertitude.

- Le premier ensemble d'axiomes évoque la préférence de résultats dans l'incertain. Ces axiomes assurent un ordre de préférence sur les résultats en utilisant une fonction de valeur $V(x)$. À chaque résultat correspond une valeur scalaire. La préférence de l'agent ira vers la plus grande valeur.
- Un second ensemble d'axiomes assurent un ordre relatif dans le cas de résultats multiples dans une situation incertaine. Cette fois à chaque couple résultat (x) et décision (d) est associé une valeur scalaire en utilisant une fonction scalaire d'utilité $U(x, d)$. Lorsqu'il existe une incertitude sur le résultat x, les décisions préférées d sont celles qui maximisent l'espérance d'utilité $E[U(x, d)|\xi]$ selon la probabilité de distribution de x et étant données les informations disponibles (ξ).

Grâce à l'ensemble des principes de la théorie de l'utilité, l'acteur doit pouvoir être capable de l'exploiter c'est le *critère de rationalité*. Le critère de rationalité peut être décrit de la façon suivante [Horvitz *et al.*, 1988] :

> Soit un ensemble de préférences exprimées par une fonction d'utilité, des croyances exprimées par des distributions de probabilité, et un ensemble de décisions possibles. Un agent doit

> choisir la conduite qui maximise l'espérance d'utilité : c'est le *principe d'utilité maximale.*

Ainsi, à partir de préférences exprimées pour des composants simples, le principe permet de calculer des préférences pour des combinaisons de résultats complexes et incertains. Par conséquent dans des situations complexes, cet outil peut-être utilisé en décomposant le problème en choix plus simples.

4.2.2 Raisonnement à base de logiques mathématiques

Dans un monde dynamique, un agent est obligé de raisonner, d'inférer et de prendre des décisions qui dépendent des résultats d'actions. Le mot *raisonnement* implique des fondements théoriques venant de la logique. La logique mathématique permet de formaliser le raisonnement humain construit en utilisant des règles acceptées de tous (axiome). La logique apporte une syntaxe, c'est-à-dire, la façon de représenter des idées et le raisonnement mais également une sémantique pour donner aux formules ainsi qu'aux symboles une signification. Dans une logique mathématiques (logique des propositions, logiques de prédicats, . . .), il est possible de distinguer trois moyens de construire un raisonnement :
– raisonnement par déduction ;
– raisonnement par induction ;
– raisonnement par abduction ;
Cependant, dans le domaine qui nous intéresse ici, la modélisation et la simulation de la décision, une notion importante demeure absente dans les logiques dites « classiques » : l'*Action*. Le raisonnement se situe sur ce que fait le système avec l'ajout d'une notion de transition entre états : *Effet d'action*. L'étude logique de la représentation de la connaissance concernant un monde dynamique, c'est-à-dire la connaissance des actions et leur effet est appelé « raisonnement sur les actions ».
Plusieurs formalismes logiques pour raisonner sur les actions existent Calcul des situations (McCarthy & Hayes, 1969), Logique dynamique (Pratt, 1976), Logique Linéaire (J.-Y. Girard, 1986). Parmi les différents formalismes pour raisonner sur les actions, nous allons présenter : le *calcul des situations* (*Situation Calculus*).
Le calcul des situations est un langage de 1er ordre, crée par McCarthy et Hayes, conçu pour la représentation de mondes dynamiques. Tous les changements du monde sont le résultat d'*actions* connues. Une *situation* est un terme de 1er ordre qui décrit l'état du monde à un instant donné. La constante S_0 marque la situation initiale où aucune action n'a été déclenchée. Le symbole *do* est une fonction binaire $Sit \rightarrow Sit$, $do(a, s)$ décrit la situation qui suit s correspondant à l'exécution de l'action a. Une action peut comporter des paramètres. Par exemple, $put(x, y)$ décrit l'action de poser l'objet x sur l'objet y, dans ce cas $do(put(A, B), s)$ décrit la situation résultant du placement de A sur B si le monde se trouve dans l'état s. Les *fluents* définit des faits dans une situation. Dans le calcul des situations les actions sont représentés par des fonctions et les situations par des termes. Par exemple : $do(putdown(A), do(walk(P), do(pickUp(A), S_0)))$ décrit la situa-

tion suite à l'exécution des actions $[pickUp(A), walk(P), putDown(A)]$. Il faut distinguer le fluent propositionnel, qui retourne une valeur booléenne avec comme dernier argument un terme de situation s, du fluent situationnel qui décrit les changement d'une situation à une autre situation. Une action est spécifiée en fournissant certains axiomes. Il est tout d'abord nécessaire de savoir quelles sont les conditions pour qu'une action soit possible en fournissant un axiome de *précondition*. Pour cela il existe un prédicat spécial pour représenter des actions gardées :$Poss(a, s)$ décrit que l'action peut être réalisable dans la situation s. Pour l'exemple 4.1, il est dit que l'action $pickup(x)$, un agent peut saisir un objet x est possible si et seulement si l'agent n'est pas en train de tenir quelque chose dans la situation s et si x est posé à coté dans s et si x n'est pas trop lourd.

$$Poss(pickup(x), s) \equiv \forall x. \neg Holding(x, s) \wedge NextTo(x, s) \wedge \neg Heavy(x) \tag{4.1}$$

Il faut également spécifier comment une action affecte l'état du monde, en définissant un axiome d'effet. L'exemple 4.2 définit que laisser tombé un objet x va le casser si x est fragile. Un axiome d'effet définit les lois causales d'un domaine d'application.

$$Fragile(x, s) \supset Broken(x, do(drop(x, s))) \tag{4.2}$$

Ces axiomes sont insuffisantes si l'on veut raisonner sur les changements. Nous devons rajouter les axiomes qui définissent les fluents qui sont invariants aux actions. L'exemple 4.3 precise que l'action $drop$ ne va pas changer la couleur de l'objet.

$$colour(y, s) = C \supset color(y, do(drop(x, s))) = C \tag{4.3}$$

Cette problèmatique a été exposée comme le *problème du décors* (ou cadre), pour la représentation de ce qui ne change pas. Deux autres problèmes de bon sens apparaît : le *problème de la quantification*, comment raisonner dans un monde qui n'est pas complètement décrit ? et le *problème des ramifications*, comment prendre en compte les effets indirects des actions ? Des réponses a ces problèmes sont décritent dans [Castilho, 1998].

Des langages de programmation logique ont été développés en utilisant le calcul des situations, nous pouvons citer *Golog* [Levesque *et al.*, 1997] et *Congolog* [De Giacomo *et al.*, 2000].

4.2.3 Décision séquentielle dans l'incertain

Ce type de problème se définit de façon très général [Garcia *et al.*, 2002] comme :

> Étant donné une description (complète ou non) de l'état initial du monde et un ensemble d'objectifs à atteindre, il s'agit de trouver une suite d'actions dont l'exécution parviendra « au mieux » à réaliser les objectifs

Un *état* est la description du système à un instant donné, l'instant initial est noté s_0. Une *action* est destinée à faire évoluer l'état du système. Ces actions sont exécutées par l'agent, qui à chaque instant, décide de lancer une certaine action et le système évolue en conséquence.

C'est pour pouvoir exprimer des problèmes de décision séquentielle avec action stochastiques et observations que les processus décisionnels markoviens (PDM) ont été développés vers la fin des années 50 dans le cadre de la recherche opérationnel.Depuis les années 90, la communauté de l'intelligence artificielle utilise les PDM, particulièrement en planification, apprentissage et contrôle.

Les propriétés mathématiques des modèles décisionnels de Markov reposent sur le respect (ou non) de la propriété de Markov. De manière générale, si l'on considère l'évolution d'un système au temps $t+1$ après avoir effectué l'action a_t, la probabilité d'être dans l'état s' avec une récompense r dépend de l'ensemble des couples (état, action) passés. Quelque soit s', r et toutes les valeurs possibles des évènements passés $s_t, a_t, r_t, s_{t-1}, a_{t-1}, \ldots, r_1, s_0, a_0$, l'évolution de l'environnement s'exprime de la façon suivante :

$$P(s_{t+1} = s', r_{t+1} = r | s_t, a_t, r_t, s_{t-1}, a_{t-1}, \ldots, r_1, s_0, a_0) \tag{4.4}$$

En revanche, dans un cadre vérifiant la *propriété de Markov*, l'évolution du système au temps $t+1$ ne dépend que de l'état et de l'action à l'état t, l'évolution du système s'exprime formellement :

$$\forall (s', r, s_t, a_t) : P(S_{t+1} = s', r_{t+1} = r | s_t, a_t) \tag{4.5}$$

Les états du système sont dits markoviens, si et seulement si l'équation 4.4 est égale à l'équation 4.5.
Les processus décisionnels de Markov s'appliquent pour un espace d'actions et d'états finis. Formellement un MDP est défini par (S, A, T, R) :
- $S = \{s\}$: un ensemble d'états fini ;
- $A = \{a\}$: un ensemble d'actions fini ;
- $T : S \times A \rightarrow \mathcal{P}(S)$: Une fonction de transitions d'états. $T(s, a, s') = \mathcal{P}(s_{t+1} = s' | s_t = s, a_t = a) \forall s, a, s'$ fait correspondre à chaque s', s, a une probabilité de transition ;
- $R : S \times A \times S \rightarrow \mathcal{P}(\mathbb{R})$: Une fonction de récompense qui à une transition associe une distribution de probabilité. $R(s, a, s') = E\{r_{t+1} | s_t = s, a_t = a, s_{t+1} = s'\}$ est la valeur de la récompense attendue.

Les processus décisionnels de Markov permettent de modéliser la dynamique d'un système soumis au contrôle d'un agent dans un environnement stochastique. On note une politique π, qui définit quel action l'agent choisit à chaque instant.

Définition. *(Politiques markoviennes déterministes stationnaires) Une politique est une application :* $\pi : s \in S \rightarrow \pi(s) \in A$

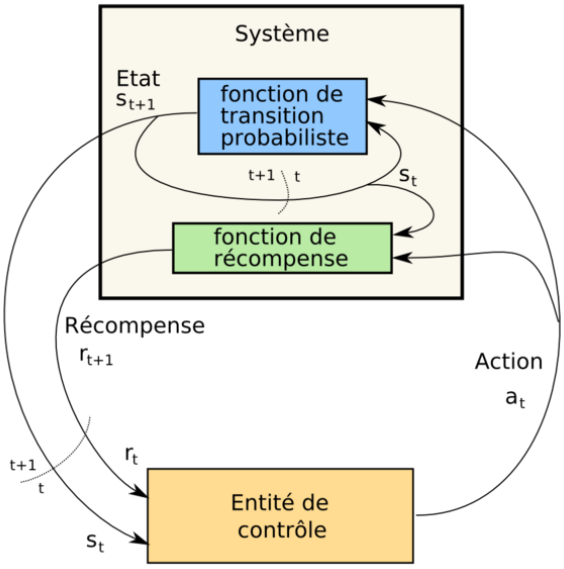

FIG. 4.3: *Processus de décision markovien*

4.2.4 Approche multi-agents

Comme présenté dans les paragraphes précédents, les recherches pour concevoir des systèmes intelligents se sont concentrées sur la représentation du langage et l'Intelligence Artificielle symbolique.

Des limites à cette vision symbolique ont été relevées par Bickard et Terveen (93a, 94) qui ramènent l'essence du cognitivisme de l'IA symbolique à une notion « d'encodage » également appelé : « Problème de l'ancrage des symboles ». Un modèle IA classique manipule des structures de symboles qui sont dénuées de sens pour le système, il ne reconnaît que leur forme (la sémantique est attribuée par l'extérieur). Peut-on réduire la cognition à la manipulation de symboles fondées sur leur forme ? Cette discussion est toujours d'actualité, périodiquement des articles relancent le débat[3].

Les difficultés évoquées précédemment ont éveillé l'intérêt des chercheurs pour de nouvelles approches en particulier l'approche *Multi-Agents*. Les systèmes multi-agents (SMA) sont reconnus comme un domaine de recherche à part entière ([Frankelin and Graesser, 1996]) depuis le milieu des années 90 avec des axes de recherches partagés par les réseaux, la recherche opérationnelle, la biologie du comportement, les sciences cognitives, etc

L'introduction de l'ouvrage *Multi-Agent Systems* [Wooldridge, 2002] de **M. Wooldridge** évoque l'IA comme un sous-domaine des systèmes multi-agents, l'Intelligence Artificielle tente de concevoir un agent « intelligent » tandis que

[3]La discussion contemporaine autour du problème de l'ancrage des symboles a été déclenchée par le "Chinese Room Problem" dans l'article "Minds, Brains ans Programs" [Searle, 1980]

les systèmes multi-agents s'intéressent à « l'intelligence » d'un ensemble d'agents.
Le concept d'*agent* dans un système multi-agents admet plusieurs définitions mais rejoignent des idées communes, nous avons choisis celle de *Wooldridge* et *Jennings* [Wooldridge and Jennings, 1995][4] :

Définition. *An agent is a computer system that is situated in some environment and that is capable of flexible autonomous action in this environment in order to meet its design objectives.*[5]

Cette définition permet de souligner les deux caractéristiques principales d'un agent :
- la capacité d'un agent d'agir *partiellement* sur son environnement à partir des entrées sensorielles qu'il reçoit de ce même environnement.
- l'autonomie d'un agent dans sa capacité à agir sans l'intervention d'un tiers et contrôler ses propres actions en plus de son état interne pour atteindre le but qui lui a été fixé.
- le terme flexible signifie que l'agent doit être capable de percevoir son environnement et d'élaborer une réponse dans le temps requis. L'agent doit exhiber un comportement pro-actif et opportuniste en prenant l'initiative au bon moment. L'interaction de l'agent avec les autres agents souligne le caractère social de ce dernier pour offrir ces compétences à la disposition des autres agents ou bien inversement faire appel à des compétences complémentaires.

Deux visions s'opposent dans la conception des systèmes multi-agents : on distingue les systèmes multi-agents dits « cognitifs » et les systèmes multi-agents réactifs. Les SMA cognitifs s'appuient sur l'intelligence des agents de manière individuelle avec un nombre réduit d'agents pouvant effectuer un grand nombre de traitements. L'autre vision s'appuie sur l'intelligence collective émergente d'un grand nombre d'agents doués de faibles connaissances individuelles.

4.3 Simulation d'un processus décisionnel

Nous pouvons nous demander, quelle sont les domaines qui s'intéressent à la simulation de la décision ? Nous pouvons citer immédiatement le domaine de la robotique pour le développement de systèmes autonomes. Nous nous sommes intéressé aux jeux de stratégies temps-réels (RTS) pour la gestion des ressources dans un temps limité. L'article [Chan *et al.*, 2007a] et [Chan *et al.*, 2007b] décrivent des algorithmes pour la planification de production de ressources dans un temps limité. Ces algorithmes ont été implémentés dans le moteur de stratégie : **Wargus**.

Le planificateur est décomposé en 2 parties :

[4] le lecteur intéressé par ce sujet pourra se référer à la bibliographie en fin du rapport

[5] Un agent est un programme informatique situé dans des environnements, qui a la capacité de mener des actions de façon autonome et flexible dans cet environnement pour atteindre ses objectifs fixés à sa conception.

– un planificateur séquentiel qui cherche un plan à partir de l'état du jeu S pour atteindre le but G en un minimum d'actions par une analyse des fins moyens (MEA).
– Un ordonnanceur qui réordonne les actions dans un plan séquentiel en gérant la concurrence pour réduire les délais.

4.4 Approches DEVS d'un processus décisionnel

Après une introduction au formalisme DEVS, nous allons nous intéresser aux travaux se rapportant à l'utilisation du formalisme DEVS dans un système décisionnel.

4.4.1 Le formalisme DEVS

DEVS (abréviation de *Discret EVent system Specification*) est un formalisme modulaire et hiérarchique destiné à la modélisation et à la simulation de systèmes à événements discrets. Ce formalisme [Zeigler, 1976] a été proposé publiquement par Bernard P. Zeigler en 1976 dans son ouvrage : *Theory of Modeling and Simulation.*

Définition d'un modèle atomique

Un modèle atomique DEVS se décrit formellement comme un 7-uplet :
$M =< X, S, Y, \delta_{int}, \delta_{ext}, \lambda, ta >$
X est l'ensemble des valeurs d'entrées
S est l'ensemble des états
Y est l'ensemble des valeurs de sorties
δ_{int} : $S \rightarrow S$ définit la fonction de transition interne
δ_{ext} : $Q \times X \rightarrow S$ définit la fonction de transition externe
λ : $S \rightarrow Y$ définit la fonction de sortie
ta : $S \rightarrow \mathbb{R}_0^+ \cup \infty$, fonction d'avancement du temps
Q = $\{(s, e)|s \in S, 0 = e = ta(s)\}$ définit l'ensemble de tous les états

Définition d'un modèle couplé

Un modèle couplé (ou réseau de modèles) est composé de plusieurs modèles atomiques ou couplés : forme une structure mathématique composé uniquement de variables, définit formellement :
$M =< X, Y, D, \{Eic\}, \{Eoc\}, \{Ic\} >$
X est l'ensemble des ports d'entrées et des valeurs associées
Y est l'ensemble des ports de sorties et des valeurs associées
D est l'ensemble des identifiants des sous-modèles avec $(M_d|d \in D)$
Eic est l'ensemble des connexions d'entrées
Eoc est l'ensemble des connexions de sorties
Ic est l'ensemble des connexions internes

Composition hiérarchique de modèles

C'est en utilisant la définition d'un modèle couplé que l'on voit apparaître le concept de hiérarchie de modèles. La théorie des systèmes décrit 2 aspects de la spécification : le niveau de spécification d'un système et le formalisme d'un système. Un système complexe peut être vu comme une décomposition ou une composition de systèmes.

- Décomposition : Comment un système peut-être divisé en composant du système ?
- Composition : Comment des composants peuvent être couplés pour construire un système ?

La théorie des systèmes se rapproche plus du concept de composition de systèmes (figure 4.4).

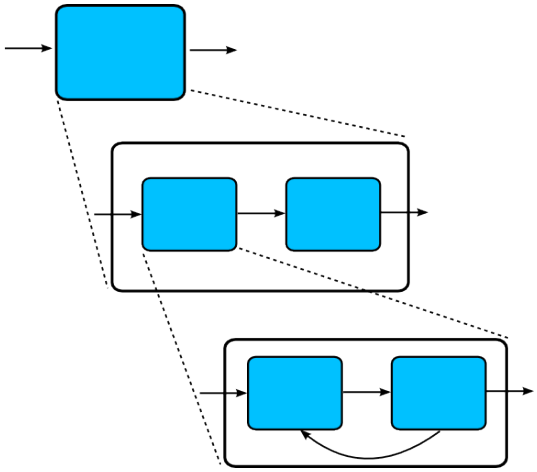

FIG. 4.4: *Composition hiérarchique de modèles*

Propriété de fermeture sous couplage

Cette propriété permet de garantir qu'un modèle DEVS couplé est rigoureusement équivalent à un modèle DEVS atomique. L'objectif est de démontrer que :

$$N =< X, Y, D, \{Eic\}, \{Eoc\}, \{Ic\} > \leftrightarrow M =< X, S, Y, \delta_{int}, \delta_{ext}, \lambda, ta > \qquad (4.6)$$

La méthode utilisée est d'écrire le modèle résultant du modèle couplé en définissant les fonctions de transition, la fonction d'avancement du temps, la fonction de sortie et l'ensemble des états.

Simulateur DS-DEVS

DS-DEVS (*Dynamic Structure Discrete Event System Specification*) est une extension proposée par F. Barros [Barros, 1997] qui permet à un modèle couplé de modifier sa structure de façon dynamique. Le concept de système à structure dynamique a été formalisé avec une preuve pour la propriété de fermeture sous couplage qui rend la composition hiérarchique de système

possible. Un modèle à structure dynamique se définit par :
$DSSN_n =< X_n, Y_n, \chi, M_\chi >$ où
– X_n et Y_n sont les ports d'entrées et de sortie du réseau
– χ le nom du modèle exécutif
– M_χ le modèle exécutif

$$M_\chi = (X_\chi, Y_\chi, S_\chi, \delta^\chi_{int}, \delta^\chi_{ext}, ta_\chi, \delta_\chi, \Sigma^*, \gamma) \tag{4.7}$$

$\gamma : S_\chi \to \Sigma^*$ définit la fonction de structure
Σ^* l'ensemble des structures
$\Sigma_\alpha \in \Sigma^* : \Sigma_\alpha = (D, Eic, Eoc, Ic)$ l'état du réseau de structures correspondant à un état du modèle exécutif

4.4.2 DEVS et la décision

Plusieurs travaux utilisent la modélisation DEVS en vue de simuler un système décisionnel, le plus souvent dans un paradigme *Multi-Agent* pour des raisons cités en 4.2.4. Nous pouvons citer les travaux de A. M. Uhrmachrer [Uhrmacher and Arnold, 1994] qui a formalisé dans *James* les SMA dans DEVS. Ces travaux mettent en relation les modèles atomiques avec les agents, qui sont modélisés comme des automates à états finis. Les caractéristiques d'un agent se retrouvent sous forme de fonctions dans DEVS :
– La fonctions de transitions internes représente l'autonomie ;
– La fonctions de transitions externes modélise la perception ;
– la fonctions de sorties représente l'action des agents sur l'environnement ou sur les autres agents.

Toujours dans le domaine des SMA, nous pouvons citer également les travaux de Sarjouhan [Sarjoughian *et al.*, 2005], mettant en place un bus d'informations (Knowledge Interchange Broker) partagé entre le formalisme RAP (Reactive Action Planner) et des modèles d'agents en DEVS. Chaque formalisme ne travaillant pas simultanément pour chaque décision, le moteur de simulation DEVS est arrêté en attendant une réponse du modèle RAP.
Des travaux, au Laboratoire d'Informatique du Littoral, sur les réseaux de Petri et DEVS ont donnés naissance à l'encapsulation dans le formalisme DEVS présenté dans la suite.

4.5 Cadre théorique du Réseau de Petri

4.5.1 Le formalisme de Réseau de Petri

Le formalisme de Réseau de Petri, inventé en 1962 par **Carl Petri**, est représenté graphiquement par un graph direct bipartit, avec deux type de nœuds (places et transition) représenté par un cercle et une barre noire. Les arcs du graphe sont classifiés par rapport au transition comme :
– *arc en entrée* : un arc fléché d'une place vers un arc
– *arc en sortie* : un arc fléché d'une transition vers un arc
– *arc inhibiteurs* : arc avec un rond d'une place vers une transition

Définition. *Un réseau de Petri généralisé non-marqué est défini comme un triplet* (S, T, A) *, où*
- S *définit l'ensemble fini non vide des places*
- T *définit l'ensemble fini non vide des transitions*
- $S \cap T = \emptyset$ [6]
- $A \subseteq (S \times T) \cup (T \times S) \rightarrow \mathbb{N}^+$ *définit un multi-ensemble d'arcs entre une place et une transition ou bien entre une transition et un arc*

Une place correspond à une *variable d'état* du système qui va être modélisé et une transition à un *événement et/ou une action* qui va entraîner l'évolution du réseau c'est-à-dire l'évolution des variables d'état du système.

Définition. *le marquage noté* M *d'un Réseau de Petri est un vecteur correspondant au nombre de jeton dans chaque place : la* i^{eme} *composante correspond au nombre de jetons dans la* i^{eme} *place. Le marquage initial, noté* M_0*, est le marquage à l'instant initial,* $t = 0$*.*

Définition. *Un Réseau marqué est un doublet* $< Q, M_0 >$ *où* Q *est un réseau de Petri non marqué et* M_0 *le marquage initiale.*

Une transition T_j est franchissable si $\forall P_i \in T_j, M(P_i) \equiv Pre(P_i, T_j)$ c'est-à-dire que pour tout place P_i en entrée d'une transition T_j, la transition T_j sera franchissable si le nombre de jetons présent dans chaque place P_i est supérieur ou égal à la valeur de l'arc reliant la place P_i à la transition T_j. Après un franchissement, nous obtenons un nouveau marquage $M\prime$: $\forall P_i \in P, M\prime(P_i) = M(P_i) - Pre(P_i, T_j) + Post(P_i, P_j)$.

4.5.2 Cadre formel du réseau de Petri en DEVS

Le modèle DEVS du réseau de Petri implémenté dans VLE est une encapsulation (*wrapping*) du formalisme de réseau de Petri. Le modèle DEVS est une implémentation d'un réseau de Petri *généralisé*. Plusieurs méthodes existent pour développer un simulateur de réseau de Petri , classiquement nous retrouvons les méthodes objets et les représentations matricielles [C. Jacques, 2002]. L'extension développée s'oriente vers une définition objet avec deux autres caractéristiques qui étend la sémantique d'un réseau de Petri [Balbo *et al.*, 2002] [Hass, 2002] :
- P-temporisé et T-temporisé : le passage d'un jeton sur une place ou une transition est temporisé au lieu d'être immédiat. Le simulateur DEVS reprend la main à chaque délai
- Priorité : dans une configuration où une place est relié à plusieurs transition, le ou les transitions iront vers la transition dont la priorité est la plus faible[7]
- Stochastique : une probabilité de franchissement est attachée à chaque transition selon une loi uniforme

La description DEVS d'un réseau de Petri se fait en spécifiant le graphe du réseau et les places ou transitions accessibles de l'extérieur. Un événement

[6] Un objet ne peut pas être une place et une transition
[7] comme pour les processus dans un système UNIX

sur ce port se traduit dans le modèle du réseau de Petri par la création d'un jeton. Tout cela se décrit dans les conditions d'une expérience du ficher de simulation vpz. Le composant comporte quatre ports :

- $places$: l'ensemble des places du réseau décrit par le nom de la place, le nom du port associé à la place et un délai si la place est temporisée.
- $transitions$: l'ensemble des transitions décrit par le nom d'une transition et si la transition est une entrée (*input*) ou sortie (*output*), le nom du port associé à la transition, le délai si la transition est temporisée et un nombre associé à la priorité.
- $arcs$: l'ensemble des arcs avec la place ou la transition d'origine, la place ou la transition destination et le nombre de jetons de l'arc sinon 1.
- $initialMarkings$: l'ensemble du marquage initial décrit par le nom de la place et le nombre de jetons.

De nombreuses applications utilisent le formalisme du réseau de Petri notamment pour la modélisation de supervision des systèmes discrets, ou modéliser des activités parallèles ou en concurrence.
Le papier [Guan *et al.*, 2008] décrit l'utilisation de réseau de Petri Hybrides (utilisation de nœuds discrets ou continues) pour modéliser les activités agricoles pour une culture de cannes à sucre avec une gestion des ressources (travailleurs et engins agricole). La thèse [Bakam Tchiakam, 2003] décrit l'utilisation des réseaux de Petri colorés pour la résolution des problèmes de gestion de ressources renouvelables, associé à un langage de spécification, validation et vérification formelle de réseaux de Petri.

CHAPITRE 5

Cas d'étude : le modèle SACADEAU

En partant du modèle de simulation SACADEAU, qui existe sous une version implémentée, nous allons en proposer une réécriture dans le formalisme DEVS des processus biophysiques et des processus de décisions. Le modèle associé aux processus de décisions fera intervenir un formalisme de plus haut niveau : le cadre théorique des réseaux de Petri, décrit dans le chapitre précédent. Quant aux processus biophysiques, nous avons utilisé un modèle plus simple et adéquat à notre problématique. Dans ce chapitre, nous allons tout d'abord décrire le modèle SACADEAU en justifiant les caractéristiques intéressantes de ce modèle avant de nous intéresser à sa réécriture. Pour finir, nous décrirons la mise en œuvre du modèle DEVS chargé du processus de traduction d'une spécification décrivant un ou des itinéraires technique en réseau de Petri.

5.1 Le projet SACADEAU

Dans cette partie, nous décrirons le modèle SACADEAU. Nous allons décrire à travers la modélisation des processus biophysiques et des processus de décisions des agriculteurs les particularités intéressantes du modèle qui a influencé dans le choix du système décisionnel à formaliser en DEVS.

5.1.1 Description du projet

Le projet **SACADEAU** (Système d'Acquisition de Connaissance pour l'Aide à la Décision pour la qualité de l'EAU) vise à développer une démarche et des outils aux personnes en charge d'un territoire dans un objectif de maîtrise de la qualité de l'eau. Le projet s'est consacré au développement d'un modèle de simulation des pratiques de désherbage des cultures de maïs et du transfert d'herbicides qui en résulte et de l'apprentissage de relations. Cette simulation permet d'évaluer les impacts en terme de qualité de l'eau dans le bassin versant du Frémeur, dans le Morbihan [Tortrat, 2005]. Un

bassin versant représente le territoire drainé par une rivière en amont d'un point nommé l'exutoire. Il est possible, avec le simulateur, de tester différents *scénarios* (voir figure 5.1) afin d'évaluer l'impact de stratégies de désherbage, en fonction de la variabilité climatique et de l'aménagement spatiale du territoire. Dans les travaux de [Trépos, 2008], les résultats

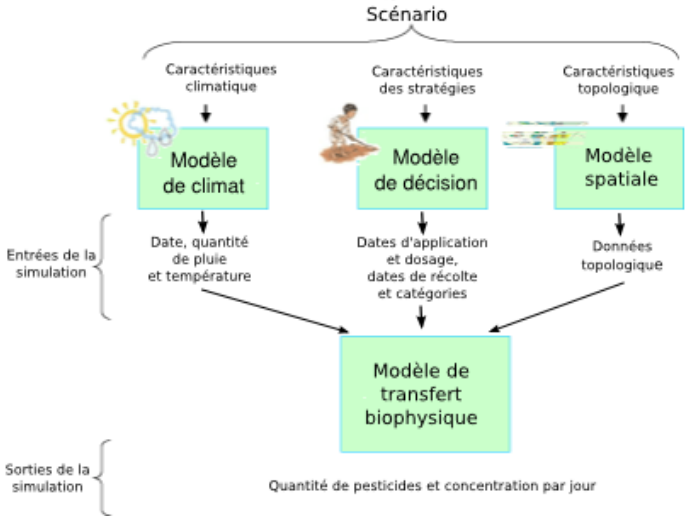

FIG. 5.1: *Modèle SACADEAU [Cordier et al., 2005]*

des simulations sont exploités pour rechercher, par apprentissage symbolique, les variables explicatives des phénomènes observés ou simulés afin de mieux comprendre les facteurs importants des processus de transferts d'herbicides.

5.1.2 Modélisation des processus biophysiques

La modélisation des processus biophysiques s'attache à décrire les processus d'écoulements durant un transfert d'eau et d'éléments chimiques. La modélisation des écoulements dans SACADEAU s'appuie sur des spatialisations différentes de la surface du sol et du bassin versant : l'un pour le ruissellement et l'autre aux écoulements de subsurface (voir figure 5.2).

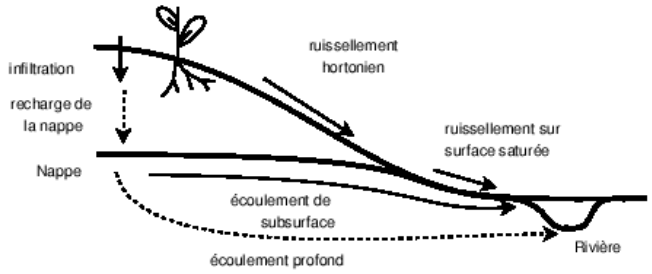

FIG. 5.2: *Différents écoulements dans un bassin versant [Trépos, 2008]*

5.1.3 Modélisation des processus décisionnels

La modélisation des processus décisionnels doit permettre de simuler différents itinéraires techniques de désherbages avec des produits, doses et à des dates variables pour chacune des parcelles cultivées en maïs. Voici une définition de la stratégie d'application de désherbage [Trépos, 2008] (ou itinéraire technique de désherbage) :

Définition. *Un itinéraire technique de désherbage définit une stratégie d'application conditionnant les fenêtres temporelles d'application du pesticide avec les matière actives utilisées avec la classe de risque de transfert associée (fort, moyen, faible).*

- *Une stratégie de type « pré levée » puis « post levée » consiste en deux applications, la première juste après le semis du maïs et la seconde au stade 5 feuilles du maïs.*
- *Une stratégie de type « tout en post levée » consiste aussi en deux applications, la première au stade 3 feuilles du maïs et la seconde au stade 5-7 feuilles.*

L'utilisation des fenêtres temporelles d'actions prédéfinies selon les pratiques agricoles signifie que le déclenchement d'une action est contraint de se situer à l'intérieur des bornes temporelles définies. D'autres contraintes viennent s'ajouter au modèle décisionnel, en particulier des contraintes liées au temps de travail et à la disponibilité des machines. Ces contraintes vont avoir pour effet l'étalement des actions dans le temps, sur la fenêtre temporelle prédéfinie, et dans l'espace. L'aléa climatique est également un paramètre décisif pour l'activation ou non d'une action. En cas de report, les actions dans l'ensemble du bassin seront soit étalées ou au contraire concentrées sur l'échelle du temps.
Le modèle décisionnel est couplé au modèle biophysique de transfert par conséquent l'emplacement géographique d'une parcelle est également un facteur déterminant pour effectuer un acte technique. En effet une parcelle en bas du bassin mettra plus longtemps à s'assécher contrairement à une parcelle en haut du bassin.
Toutes ces contraintes seront exprimées formellement à l'aide de règles de décision qui détermine partiellement le déclenchement d'une action. Les règles de décision se présentent sous la forme : $C \rightarrow A$, l'action A est déclenchée si les conditions C sont satisfaites[1].
Dans le bassin, les agriculteurs se regroupent en coopérative se qui permet de partager un nombre fixe de machines agricoles qui comme dit précédemment conditionne fortement la faisabilité d'une activité culturale. Le fonctionnement détaillé du modèle SACADEAU est décrit dans la thèse [Trépos, 2008].

[1] On notera que l'opérateur $\rightarrow$ n'est pas un connecteur logique mais une transition d'état (voir paragraphe sur logique)

Exemple de règle de décision du modèle décisionnel de SACADEAU

Une application de pesticide peut-être exécutée un jour J sur une parcelle P si les conditions suivantes sont remplies :
- J est contenu dans la fenêtre temporelle prévue par l'itinéraire technique pour cette application.
- La quantité de pluie au jour J est inférieure à un seuil prédéfini
- depuis le dernier jour de pluie, le temps écoulé est suffisant pour un réessuyage du sol assurant une condition de portance du sol et permettant aux engins agricoles de rentrer sur la parcelle P
- le temps nécessaire au traitement de la parcelle P (lié à la surface de P) ne doit pas trop dépasser la durée du temps de travail journalier de l'exploitant
- Un engin agricole est à disposition de l'exploitant pour l'application.

5.2 Présentation des choix généraux

Nous avons choisis de modéliser et de simuler les pratiques agricoles décrites dans un itinéraire technique, destiné au pilotage d'une production agricole constituée de plusieurs parcelles (voir figure 5.3). Le système ressemble aux modèles du projet SACADEAU, nous retrouvons les deux majeures parties : le modèle destiné à représenter les processus *biophysiques* et le modèle *décisionnel*. Les caractéristiques du modèle décisionnel de SACADEAU ont été décisif dans le choix du système à formaliser en DEVS. Ce modèle n'est pas constitué uniquement de règles de décisions mais nous y trouvons des contraintes supplémentaires qui augmentent la complexité du système décisionnel : le parallélisme dans la gestion simultanée de plusieurs parcelles, une gestion des ressources pour l'utilisation d'engins agricole pouvant être utilisé sur une seule parcelle à la fois, l'utilisation de fenêtres temporelles associées à une activité culturale.

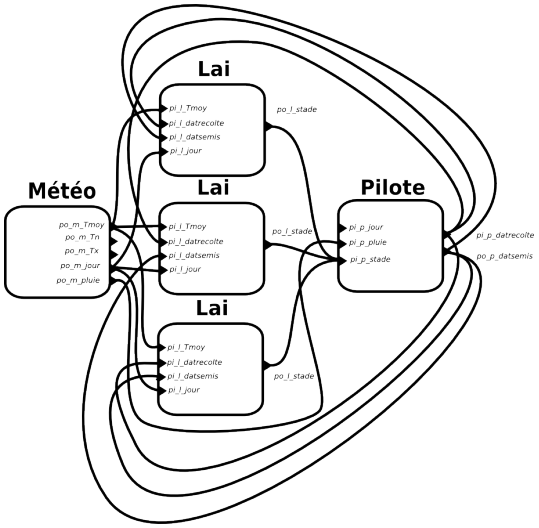

FIG. 5.3: *Modèle de simulation des pratiques agricoles*

La conception du modèle décisionnel ressemble au modèle décisionnel de SACADEAU présenté dans le paragraphe 5.1.3. Nous retrouvons le concept d'itinéraire techniques qui décrit les actions à mener sur le modèle biophysique associé. Nous allons tout d'abord décrire le modèle biophysique sur lequel se repose le modèle décisionnel, puis décrire ce dernier en expliquant plus en détail sa structure.

5.3 Le modèle biophysique en DEVS

Le modèle biophysique a été simplifié par rapport aux modèles biophysiques du projet SACADEAU, adapté d'un modèle existant développé dans le projet *RECORD*, pour modéliser la croissance d'une plante de maïs. La modélisation des systèmes hydrographiques peut aisément être ajoutée par la suite sans conséquence pour le modèle décisionnel.

Modèle de génération de Météo

La croissance de la plante est dépendante de la météo, plus particulièrement de la pluie et la température. Le modèle de plante est par conséquent perturbé par un modèle générateur de *Météo*. Le modèle *Météo* utilisé génère des événements à partir des informations lues dans un fichier météo, à pas de temps journalier, la température moyenne, minimal, maximale et la hauteur des précipitations. Le modèle atomique ne comporte pas de port d'entrée et cinq ports de sortie :
– po_m_pluie : la quantité de pluie tombée le jour j
– po_m_Tx : la température maximale du jour j
– po_m_Tn : la température minimale du jour j
– po_m_Tmoy : la température moyenne du jour j
– po_m_jour : la date au format jour julien $(1 \ldots 365)$[2]
C'est le modèle *Météo* qui dirige la durée maximale de la simulation fixée à 365 unité de temps (jours).

Modèle de culture

Le modèle de culture utilise un modèle générique basé sur l'évolution de l'indice foliaire : *Leaf Area Index* (LAI). L'indice foliaire est la surface de feuilles par m^2 de sol. L'indice évolue de façon linéaire avec la somme de température reçue par la plante (voir figure 5.4). La température moyenne quotidienne (Tmoy) est calculée par le modèle *Météo* comme la moyenne entre la température minimale et maximale du jour j. La somme des températures correspond au cumul des températures moyennes quotidiennes depuis la date de semis (*CumST*). Le modèle est considéré dynamique, discret à pas de temps journalier, il se formalise par des équations aux différences :

$$LAI(t+1) = LAI(t) + \delta(t) \tag{5.1}$$

[2]À la date $t = 0$, le modèle météo fournit intentionnellement des valeurs à zéro et commence à lire les données météo à $t = 1$ ce qui permet de synchroniser tout le modèle

avec :
– Pour $CumST(t) < ST1 : \delta(t) = 0$
– Pour $ST1 < CumST(t) < ST2 : \delta(t) = f_{ST1-ST2}(Tmoy(t))$
– Pour $ST2 < CumST(t) < ST3 : \delta(t) = 0$
– Pour $ST3 < CumST(t) < ST4 : \delta(t) = f_{ST3-ST4}(Tmoy(t))$
– Pour $ST4 < CumST(t) : \delta(t) = 0$

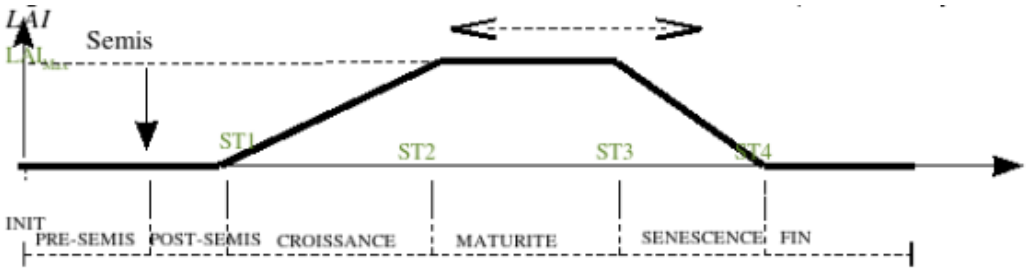

FIG. 5.4: *Évolution de l'indice foliaire*

C'est le couplage du modèle *Météo* et du modèle de culture qui forme le modèle biophysique (voir figure 5.5). Le modèle de culture comporte quatre ports d'entrées
– pi_l_jour : la date du jour j
– pi_l_Tmoy : la température moyenne du jour j
– $pi\_l\_datsemis$: la date du semis
– $pi\_l\_datrecolte$: la date de récolte
et un port de sortie po_l_stade qui informe le stade de la plante parmi :

1. semis → levée
2. croissance → maturité
3. maturité → sénescence
4. sénescence → mort

Le modèle de culture peut fonctionner selon deux modes de fonctionnement : un mode *autonome* et un mode *piloté*. Le mode *autonome* permet simplement d'utiliser le modèle avec un générateur de météo en précisant la date de semis et la date de récolte dans les conditions d'expérience. L'autre mode permet de spécifier les mêmes dates (date de récolte et date de semis) sous la forme d'événements envoyés pas un autre modèle. Le modèle se met dans un état `IDLE` pour attendre le 1reévénement correspondant à l'activité culturale de *semis*, sans cela le modèle ne fera rien.

5.4 Le modèle décisionnel en DEVS

Le modèle décisionnel utilise l'extension *DS-DEVS* (voir paragraphe 4.4.1) de *VLE* qui lui permet de manipuler sa propre structure (connexion et/ou port) de façon dynamique afin de s'adapter au nombre de parcelles à piloter. Le processus décisionnel se retrouve séparé en deux modèles atomiques : le modèle *pilote* et le modèle DEVS encapsulant le réseau de Petri. Le modèle

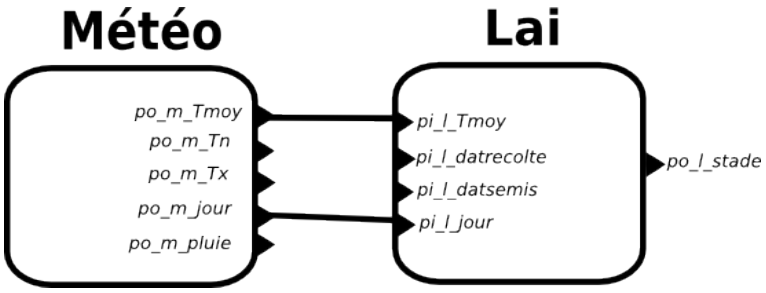

FIG. 5.5: *Modèle couplé météo-lai*

du réseau de Petri aura pour rôle de représenter les itinéraires techniques des parcelles à piloter. Un troisième modèle appelé *Traducteur* est utilisé uniquement à l'initialisation du système pour la création des modèles de culture et du modèle du réseau de Petri en suivant la description de la simulation. La séparation du modèle décisionnel en deux parties est liée à un choix d'implémentation qui se justifie par le souhait d'avoir une séparation entre l'itinéraire technique et les différentes règles de décision associés à une activité culturale.

Le modèle du pilote

Nous pouvons comparer le modèle de pilote comme un contrôleur de modèles dynamiques. Le pilote contrôle le pilotage des systèmes de culture par l'intermédiaire du modèle englobant le réseau de Petri. Pour expliquer le fonctionnement du pilote, nous allons examiner son fonctionnement par les différents états du modèle DEVS (voir figure 5.6).

Nous distinguons cinq transitions, à chaque état nous avons la durée dans laquelle le modèle reste dans l'état :
- `INIT` : l'état consacré à l'initialisation du modèle, ce qui englobe des opérations de traduction décrit dans la suite du document
- `IDLE` : c'est dans cet état que le pilote bascule immédiatement après avoir passé l'étape l'initialisation du modèle pour attendre des événements en provenance du modèle météo.
- `UPDATE` : à chaque pas de temps, cet état consiste à identifier pour chaque parcelle, les activités pouvant être effectuées.
- `EXEC` : cet état correspond à l'envoi d'événements vers le réseau de Petri pour signifier que des activités peuvent être activées.
- `STOP` : cet état est lié à l'état `EXEC`, dans le cas où le modèle a notifié le réseau de Petri. Si une activité ne pas s'effectuer, le pilote est prévenu pour ne pas considérer une activité comme effectuée.

Le composant du pilote comporte avant l'étape d'initialisation 3 ports d'entrée :
- pi_p_jour : la date du jour j
- pi_p_stade : stade des modèles de culture[3]
- pi_p_pluie : quantité de pluie tombée le jour j

[3]un seul port d'entrée quelque soit le nombre de modèles de culture

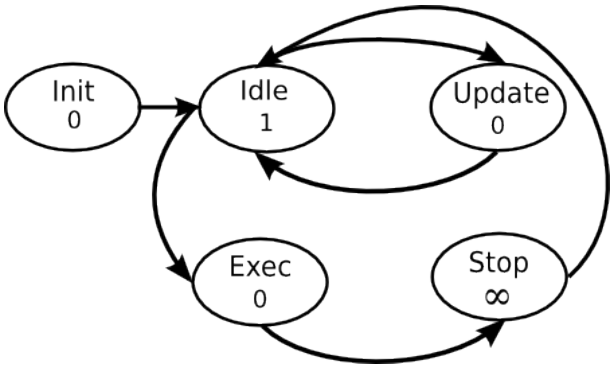

FIG. 5.6: *Graphe d'états du pilote*

et de 2 ports de sortie :
- po_p_nblai : nombre de modèles de culture à créer par le modèle du Traducteur
- po_p_usepn : booléen pour indique la création du réseau de Petri

Le modèle du pilote peut être utilisé sans le réseau de Petri ce qui peu d'intérêt puisqu'il n'y aura pas de gestion des ressources. Dans ce cas le modèle est directement connecté aux modèles de culture de façon dynamique.
Nous devons maintenant nous intéresser aux données manipulées par le modèle qui n'a pas été précisé dans le graphe d'état. Deux types de données sont manipulés par le pilote : des données statistiques issues de la description de l'itinéraire technique et différents types de prédicats correspondant aux règles de décision. Les données statiques sont les activités culturales décrites dans l'itinéraire technique. Le pilote « instancie » à l'initialisation (dans l'état `INIT`) chaque activité pour chaque parcelle en l'ajoutant dans sa liste associées des activités à mener. Le lien entre la représentation d'une parcelle et le modèle de culture se fait de façon symbolique (par son nom).
Le pilote a pour rôle d'ordonnancer les activités de l'itinéraire technique selon un ordre temporel. Les activités de chaque parcelle sont triés selon la borne inférieure de la fenêtre temporelle associée. Les règles de décision de chaque activité se retrouvent traduit sous la forme de prédicat. Pour que le déclenchement d'une action se fasse, le prédicat a besoin de données de l'environnement ainsi que des données statiques. À chaque pas de temps t, le pilote « évalue » pour chaque parcelle les activités déclenchables. Pour savoir si une activité est à effectuer, il suffit d'évaluer les différents prédicats qui doivent tous renvoyer la valeur booléenne `VRAI`.
Une activité comporte deux types de prédicats :
- un prédicat lié à la pertinence d'activation, par exemple savoir si t se trouve dans l'intervalle temporelle.
- un prédicat lié à la faisabilité d'une activité culturale, par exemple la portance du sol

Nous pouvons assimiler une activité au concept d'action, par conséquent l'exécution d'une action a un effet dans le monde qui se traduit dans le formalisme DEVS par un événement. Dans le cas où une activité respecte toutes les conditions, le pilote passe de l'état `UPDATE` dans l'état `EXEC` et

un événement est envoyé au modèle du réseau de Petri. Le réseau de Petri permet entre autre de gérer les contraintes sur les ressources, c'est ce modèle qui procédera au déclenchement d'un événements vers le modèle de culture représenté par notre parcelle. Le pilote se place dans l'état `STOP`, si une activité n'a pas pu être réalisée par manque de ressources un événement est reçu pour que le pilote reconsidère l'activité lorsque le pilote évalue les activités.

le modèle traducteur

Comme précisé dans la description du modèle du pilote, le modèle du traducteur est utilisé uniquement à l'initialisation du système mais est néanmoins indispensable au fonctionnement du modèle, il correspond au modèle M_χ exécutif du modèle couplé.

Ce modèle a également 2 modes de fonctionnement comme le modèle de culture c'est-à-dire un mode *piloté* et un mode *autonome*. Le mode *autonome* permet juste de créer des modèles de cultures, les connexions entre le générateur de météo et les modèles de cultures sont gérées par le traducteur. Le nombre de modèles et les dates de semis seront précisés en début de simulation. Le mode *piloté* permet de gérer la création des modèles de cultures par le modèle du pilote. Si l'on précise l'utilisation du formalisme du réseau de Petri, le modèle permet également de traduire la spécification du réseau de Petri venant du modèle du pilote en un modèle DEVS avec la gestion des connexions entre tout les modèles atomiques. La phase de traduction sera décrit dans le paragraphe dédié à la description du langage de spécification.

La figure 5.7 illustre la création de 3 modèles de culture par le traducteur en mode piloté (la connexion avec le modèle météo a été enlevé pour une meilleure lisibilité).
Le composant comporte uniquement 2 ports d'entrée :
– $pi\_t\_nb\_model$: Nombre de modèles de culture à créer
– $pi\_t\_use\_pn$: Activer la traduction du réseau de Petri

le modèle du réseau de Petri

Nous avons utilisé le réseau de Petri pour la modélisation d'un itinéraire technique pour chaque parcelle. Le résultat du marquage du réseau permet de voir l'avancement du plan de production. Le même réseau permet de modéliser la concurrence d'accès pour les engins agricoles lorsqu'une activité en nécessite. Comme précisé dans la description du modèle du pilote si une activité ne peut pas s'effectuer, le jeton correspondant à l'événement du pilote pour effectuer l'action ne reste pas dans le réseau mais est renvoyé vers le pilote pour lui notifier. Cette notification se traduit par une priorité plus faible à la transition qui est connectée au port d'annulation de la parcelle concerné. Il est intéressant de savoir qu'une ressource

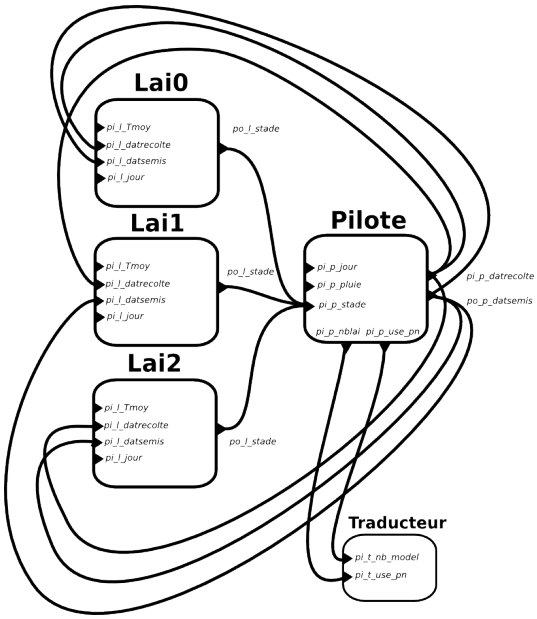

FIG. 5.7: *Mise en œuvre du traducteur en mode autonome*

consommée sera libérée à $t+1$. Le délai est intégré en ajout une transition temporisée ayant comme valeur de délai à 1.

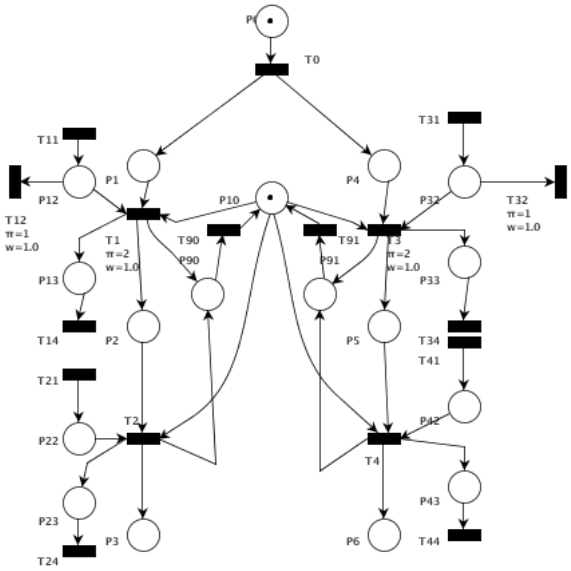

FIG. 5.8: *Réseau de Petri d'un itinéraire technique avec une ressource partagée*

CHAPITRE 6

Traducteur de réseau de Petri

Nous allons décrire dans cette partie comment se déroule la traduction d'une spécification d'un itinéraire technique à l'obtention du modèle atomique encapsulant le réseau de Petri.

6.1 Description formelle du système décisionnel

Dans un but de générer de façon automatique la création du réseau de Petri, qui se révèle inconfortable pour son usage par un modélisateur, beaucoup d'erreurs peuvent apparaître dans la création du réseau de Petri. Nous pouvons améliorer la création du réseau de Petri principalement de 2 manières : une interface graphique ou un langage spécifique (Domain Specific Language).

L'interface graphique semble une solution interactive intéressante pour décrire un plan de production cependant le développement d'interface graphique est très complexe. Il nécessite une réflexion sur l'ergonomie à adopter, la bibliothèque graphique à utiliser, et nécessairement un manuel utilisateur pour exposer la façon dont l'interface doit être utilisée.
La création d'un DSL semble une solution beaucoup plus simple et plus dans la philosophie UNIX, c'est-à-dire développer de petits outils de conception simple mais dont l'efficacité n'est pas comparable à une interface graphique. Le développement du langage n'empêche évidemment pas d'être embarqué dans une interface graphique par exemple comme une extension graphique dans *GVLE*, l'interface graphique de modélisation de VLE.

6.2 Détail de l'implémentation

Dans la famille des analyseurs lexicales et syntaxiques, il est facile de citer les outils traditionnels des systèmes UNIX ; *Flex* et *Bison*. Ces outils supportent la génération d'analyseurs lexical et syntaxique en C et également

C++ mais nécessitant du code supplémentaire entre le code généré et son utilisation pour la gestion des flux en C++. Si l'on considère, l'intégration future du langage de description dans la plateforme *RECORD/VLE* développé en C++, l'inconvénient cité précédemment font que ces outils ne sont adaptés.

6.2.1 Choix d'un analyseur syntaxique

Le choix de l'outil s'est dirigé vers une bibliothèque C++ déjà grandement utilisé dans la plateforme : *Boost*. Cette bibliothèque inclut un framework pour la génération d'analyseurs syntaxique orienté objet dénommé *Spirit*. Le framework permet facilement d'écrire une grammaire exclusivement en C++. Le support de la spécification EBNF (Extended Bachus-Normal Form) peut être aisément mélangé avec d'autre code en C++ puisque étant du C++ parfaitement valide. Grâce à la puissance des templates C++ , la grammaire est immédiatement exécutable, sans avoir besoin de la phase supplémentaire de traduction EBNF vers C ou C++.

6.2.2 Spécification de grammaire EBNF

Pour mémoire, la spécification BNF (Forme Normale de Bachus) est une notation très bien adapté à l'expression des emboîtements et de la récursivité, moins à l'expression de la répétition et du choix facultatifs. C'est la notation EBNF qui étant ces points sous la forme d'un opérateur postfixé :
- R+ indique que R peut apparaître une ou plusieurs fois, et permet d'indiquer la répétition
- R? indique que R est facultatif
- R* indique que R peut apparaître zéro, une ou plusieurs fois, et permet d'indiquer le répétition facultative

Des parenthèses peuvent être nécessaires si ces opérateurs postfixés doivent s'appliquer à plus d'un symbole.

6.2.3 L'écriture de grammaire EBNF avec Spirit

La spécification de grammaire avec l'utilisation de *Spirit* nécessite des modifications par rapport à la syntaxe EBNF original, afin d'être conforme aux règles de syntaxe C++. Parmi les modifications à effectuer par rapport à une notation EBNF, nous avons l'ajout de l'opérateur $>>$ qui correspond à la séquence. Dans la syntaxe EBNF, ab signifie b *doit être suivi de* a. Comme il n'existe pas d'opérateur 'vide' en C++ , il est tout simplement impossible de l'écrire, c'est donc l'opérateur $>>$ qui a été utilisé pour signifier *est suivi par*, il faudra donc écrire $a >> b$. L'opérateur alternatif $|$ et les parenthèses $()$ sont présents tel quels. L'opérateur d'affectation $=$ est utilisé à la place de $:=$. Pour finir l'opérateur étoile de Kleen utilisé pour marquer la répétition facultative qui est utilisé pour être un opérateur postfixé devient préfixé ainsi au lieu d'écrire en EBNF $a*$ il faut écrire $*a$. Chaque règle de grammaire étant une expression C++, elle se termine avec le traditionnel ;.

Spirit a été développé dans le but d'être un outil pratique, simple d'utilisation et modulaire. Le framework a été conçu par couches, ce qui permet d'écrire rapidement un analyseur lexical et syntaxique en ayant juste appris les concepts de base. Le framework complet est composé uniquement d'entêtes C++, sans aucune autres bibliothèques supplémentaires.

Les concepts introduit par le framework seront expliqué au fur et à mesure de la description de l'implémentation du traducteur de réseau de Petri. La description s'organisent en trois parties. La description formelle de la grammaire accepté par le traducteur. Ensuite une partie sera consacrée à la description de l'analyseur syntaxique en décrivant les structures de données utilisée résultant de l'analyse. Puis pour finir une dernière partie expliquera l'analyse sémantique dont la génération du réseau de Petri.

6.2.4 Grammaire du traducteur

La grammaire du langage appartient à la famille des grammaire *LL(1)* c'est-à-dire un analyseur descendant dont voici une description :
sequence → *bloc bloc**
bloc → *nombloc* '{' *item* '}'
nombloc → "model" | "task" | "parcelle" | "itk"
item → *expr**
expr → '(' (nombre | *expr* | *symbol*)+ ')'
symbol → (alpha|'_'|'$') (alpha_num | '_' | '!' | '-' | '>')*
L'implémentation ne suit pas exactement cette description, la grammaire se retrouve séparée en deux parties pour simplifier l'analyse sémantique mais néanmoins rend la grammaire très flexible et peut facilement évoluer. La grammaire *translator_grammar* s'occupe des règles de production *sequence*, *bloc*, *nombloc* et *item* puis la grammaire *spec_grammar* qui s'occupe des autres règles de production soit *expr* et *symbol*. Il existe une troisième grammaire *skip_grammar* qui s'occupe de supprimer les commentaires (les lignes commençant par ';') ainsi que les sauts de lignes, ces éléments ne sont pas significatifs pour le langage.

Voici un exemple de spécification d'un itinéraire technique accepté par la grammaire :

```
; description des modeles
model {
  (meteo
    (outport jour Tmoy))
  (lai
    (inport date_semis date_recolte)
    (outport maturite))
}
; liste des activites
task {
  (semis
```

```
    (predicat enable_semis)
    (timewindow (112 128))
    (effect lai->date_semis))
  (recolte
    (predicat enable_recolte)
    (timewindow (150 200))
    (effect lai->date_recolte))
}
; liste des parcelles
parcelle {
  ;declaration de 3 parcelles
  (set p1) (set p2) (set p3)
}
; itineraire technique
itk{
  (itk1 (semis recolte))
  (set! itk1 (p1 p2 p3))
}
```

Chaque bloc permet la déclaration d'une instance d'un type d'objet correspondant au nom du bloc, par exemple *meteo* déclare un objet *modele* . Il n'existe que 2 mot-clés qui n'ont pas la même signification pour l'analyseur : **set** et **set!**. Le mot-clé **set** permet de faire une déclaration symbolique nécessaire pour l'évaluation de l'itinéraire technique, ceci permet de distinguer le symbole *p1* qui ne contient pas d'attribut au contraire du symbole *semis*. Le mot-clé **set!** permet de faire l'association entre plusieurs symboles dans l'exemple, **itk1** est associé aux symboles **p1**, **p2** et **p3**. Cette association est très importante, elle détermine le point de départ pour la traduction en un réseau de Petri. Nous remarquons que les expressions à l'intérieur d'un bloc ressemble à une syntaxe `LISP`. Le type de données pour stocker les expressions doit adopter une structure récursive de liste. C'est grâce à l'utilisation d'une bibliothèque *Boost* qu'une structure puisse être déclaré et facilement manipulable grâce au design pattern des *visiteurs*. Voici la définition du type var_t :

```
typedef make_recursive_variant<
  double,
  std::string,
  nil_,
  undef_,
  std::vector< recursive_variant_> >::type  var_t;
```

Pour déclarer un liste vide se définit simplement :

```
var_t liste = nil_();
```

Chaque bloc est stocké dans une structure $tcontext\_t$ contenant une table de hachage composé d'un nom du symbole à un entier. Le nombre correspond au décalage dans la liste des expressions définie dans le bloc.

```
std::string cname; // nom du bloc
```

```
std::vector<spec_t> dat; // liste des expressions
std::map<std::string, range_t> ctxts; // table de hashage
```

6.2.5 Analyse sémantique du langage de description

La décomposition de la grammaire en plusieurs grammaires rend cette étape de l'analyse beaucoup plus simple. Le résultat de l'analyseur peut-être utilisé par un autre analyseur. L'analyse sémantique se traduit dans la bibliothèque *Boost* par l'ajout dans la grammaire d'actions sémantique. Une action sémantique est de la forme suivante :

```
expression [action]
```

Un autre concept important apparaît dans l'ajout d'actions sémantique : *le contexte*. Une instance de l'objet contexte est crée avant l'analyse d'un non-terminal et est détruit ensuite. Un contexte permet de définir le type de retour du résultat de l'analyseur par exemple pour la règle `expression` nous aurons :

```
struct variant_closure
  : spirit::closure<variant_closure, spec_t>
{
  member1 val;
};

rule<ScannerT, variant_closure::context_t> item;
```

la structure `spec_t` est la classe englobant le type récursif `var_t`. Cette classe surcharge les opérateurs `+=` qui sont utilisés dans l'analyse sémantique pour construire une expression durant l'évaluation de l'expression.

```
struct spec_grammar :
  public grammar<spec_grammar, variant_closure::context_t>
{
  template <typename ScannerT> struct definition {
      definition(spec_grammar const& self)
      {
      top = expr [self.val = arg1];
      expr = (*empty >> ( !chlit<>('\'') >> *empty
           >> chlit<>('(')
           >> +( *empty >>
                ( number [expr.val += arg1]
                | expr [expr.val += arg1]
                | symbol [expr.val += arg1]
                ) >> *empty)
           >> ( chlit<>(')') | error_missin_paren)
           >> *empty ));
       number = real_p [number.val = arg1];
       symbol = lexeme_d
       [
```

```
            ((alpha_p | '_' | '$') >>
                 *(alnum_p | '_' | '!'| '-' | '>'))
            [ symbol.val =
              construct_<std::string>(arg1, arg2)
            ]
          ];
          ;
          empty = space_p | eol_p;
        };
        rule_t top, empty;
        var_rule_t expr;
        nb_rule_t number;
        sym_rule_t symbol;
    };
};
```

6.2.6 L'évaluation de la spécification

L'étape final du traducteur consiste à construire une expression correspondante à une déclaration d'un modèle DEVS du réseau de Petri. L'idée principal pour la construction du réseau est de construire le réseau par morceau. L'idée est similaire à YAWL [Van der Aalst and ter Hofstede, 2005], où des patterns identifiés sont réutilisés pour construire un réseau de Petri complexe. Nous avons identifié 2 motifs de réseaux de Petri et une partie initialisation :

- Le réseau de Petri associé à un itinéraire technique, chaque transition correspond à une activité culturale et chaque place a l'état du modèle de culture.
- Le réseau de Petri destiné à la réception des événements du pilote, et le retour pour notifier le pilote en cas de ressource non disponible.
- Une place et une transition initiale commune pour initialiser les itinéraires techniques dans la place précédant la première activité culturale.

L'exemple d'une spécification donné dans la partie 6.2.4 fournirait donc le résultat suivant pour une parcelle :

```
(pn_lai0
  (places
   (p_1 p_12 p_13 p_2 p_22 p_23 p_3 ) )
  (transitions
   (t_1 t_11 t_14 t_2 t_21 t_24 ) )
  (arcs
   ((p_1 t_1) (t_11 p_12) (p_12 t_1 )
    (t_1 p_13) (p_13 t_14) (p_2 t_2)
    (t_1 p_2) (t_21 p_22) (p_22 t_2 )
    (t_2 p_23) (p_23 t_24) (t_2 p_3 ))))
```

L'expression résultant de la construction du réseau de Petri sera envoyé par le modèle du pilote au modèle du Traducteur qui n'aura plus qu'à analyser

l'expression pour construire le modèle du réseau de Petri. La figure 6.1 montre un exemple de construction d'un réseau de Petri pour la gestion de trois parcelles sans gestion de ressources.

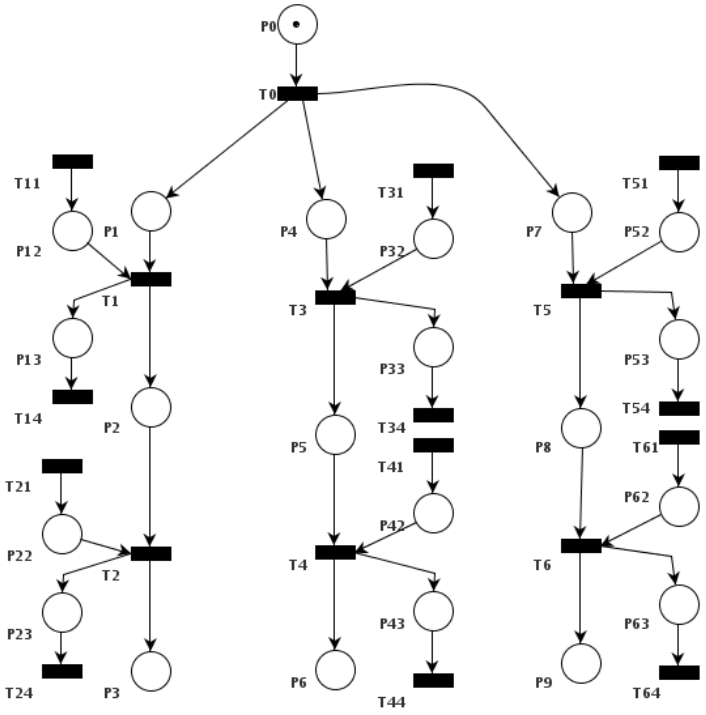

FIG. 6.1: *Réseau de Petri représentant un itinéraire technique*

Discussion

À la vue de ce travail, la faisabilité de la réécriture d'un modèle de décision en utilisant des formalismes de plus haut niveau est réalisable. Il est important de souligner que le modèle décisionnel choisi comporte des caractéristiques intéressantes, autres que uniquement des règles de décision : le parallélisme, la gestion de ressources et le concept de fenêtres temporelles.

L'utilisation du cadre de Petri, pour modéliser les itinéraires techniques à été possible. Le traducteur d'une spécification de plan de conduite vers un réseau de Petri a été intéressante pour éviter à un modélisateur de devoir écrire le réseau à la main, qui pour des plans plus complexes, avec plus d'activités ou plus de modèles de culture à gérer, aurait été difficile.

Il serait intéressant d'utiliser d'autres formalismes de la décision pouvant être géré par le traducteur par exemple les modèles décisionnels de Markov. Le langage lui-même peut également être amélioré en enrichissant la grammaire d'opérateur de composition entre activités pour la spécification de plans de production flexibles rappelant les idées de l'article [Martin-Clouaire and Rellier, 2006]. Actuellement les prédicats se retrouvent implémentés en C++, le développement du langage pourrait tenir compte de la possibilité de spécifier ces prédicats d'activation ou le support de préférence pour effectuer des tâches sur une parcelle en priorité.

Liste des figures

Bibliographie

[Bakam Tchiakam, 2003] I. BAKAM TCHIAKAM. *Des systèmes multi-agents aux réseaux de Pétri pour la gestion des ressources naturelles : Le cas de la faune à l'Est-Cameroun*. PhD thesis, Université de Yaounde 1, 2003.

[Balbo *et al.*, 2002] G. BALBO, G. CONTE, S. DONATELLI, and G. FRANCESCHINIS. *Modelling with Generalised Stochastic Petri Nets*. 2002.

[Barros, 1997] F. J. BARROS. Modeling formalisms for dynamic structure systems. *ACM Trans. Model. Comput. Simul.*, Oct. 1997.

[C. Jacques, 2002] G. Wainer C. JACQUES. Using the cd++ devs tookit to develop petri nets. In *Proceedings of the 2002 Summer Computer Simulation Conference. San Diego, U.S.A.*, 2002.

[Castilho, 1998] M. A. CASTILHO. *Modeles logiques pour le raisonnement sur les actions*. PhD thesis, Université Paul Sabatier-Toulouse III, 1998.

[Chabier *et al.*, 2007] P. CHABIER, F. GARCIA, R. MARTIN-CLOUAIRE, G. QUESNEL, and H. RAYNAL. Toward a simulation modeling platform for studying cropping systems management : the record project. *MODSIM '07*, 2007.

[Chan *et al.*, 2007a] H. CHAN, A. FERN, S. RAY, N. WILSON, and C. VENTURA. Extending online planning for resource production in real-time strategy games with search. In *Workshop on Planning in Games, (ICAPS)*, 2007.

[Chan *et al.*, 2007b] H. CHAN, A. FERN, S. RAY, N. WILSON, and C. VENTURA. Online planning for resource production in real-time strategy games. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 2007.

[Chatelin *et al.*, 2007] M.-H. CHATELIN, C. AUBRY, and F. GARCIA. A novel method for the sustainable management of wheat crops : exploration by simulation. *Agronomy for Sustainable Development*, 27(4) :337–345, Nov. 2007.

[Cordier *et al.*, 2005] M. O. CORDIER, F. GARCIA, C. GASCUEL-ODOUX, V. MASSON, J. SALMON-MONVIOLA, F. TORTRAT, and R. TREPOS. A machine learning approach for evaluating the impact of land use and mana-

gement practices on streamwater pollution by pesticides. In *MODSIM05 (International Congress on Modelling and Simulation)*, Dec. 2005.

[Crespo, 2008] O. CRESPO. *Conception par simulation pour la conduite de culture*. PhD thesis, EDSYS, 2008.

[De Giacomo *et al.*, 2000] G. DE GIACOMO, Y. LESPÉRANCE, and Levesque H.. Congolog, a concurrent programming language based on the situation calculus. *Artificial Intelligence*, 121 :109–169, 2000.

[Frankelin and Graesser, 1996] S. FRANKELIN and A. GRAESSER. Is it an agent, or just a program ? : A taxinomy for autonomous agents. In *Third International Workshop on Agent Theories, Architectures, and Languages*. Springer-Verlag, 1996.

[Garcia *et al.*, 2002] F. GARCIA, J. LANG, and A.-I. MOUADDIB. Processus décisionnels de markov. *Le temps, l'évolution et l'espace*, 2002.

[Guan *et al.*, 2008] S. GUAN, M. NAKAMURA, T. SHIKANAI, and T OKAZAKI. Hybrid petri nets modeling for farm work flow. *Computer and electronics in Agriculture*, 2008.

[Hass, 2002] P. J. HASS. *Stochasting Petri Nets : Modelling, Stability, Simulation*. Springer, 2002.

[Horvitz *et al.*, 1988] E. J. HORVITZ, J. B. BREESE, and M. HENRION. Decision theory in expert systems and artificial intelligence. In *International Journal of Approximate Reasoning*, number 2, pages 247–302, 1988.

[Kepner and Tregoe, 1997] C. H. KEPNER and B. TREGOE. *The New Rational Manager*. 1997.

[Levesque *et al.*, 1997] H. J. LEVESQUE, R. REITER, Y. LESPÉRANCE, F. LIN, and R. B. SCHERL. Golog : A logic programming language for dynamic domains. *Journal of logic Programming*, 1997.

[Martin-Clouaire and Rellier, 2003] R. MARTIN-CLOUAIRE and J.-P. RELLIER. Modélisation et simulation de la conduite d'un système de production agricole. In *4ème Conf. Francophone de Modélisation et Simulation (MOSIM'03)*, 2003.

[Martin-Clouaire and Rellier, 2006] R. MARTIN-CLOUAIRE and J.-P. RELLIER. Représentation et interprétation de plans de production flexibles. In *Journées Francophones sur la Planification, décision et l'apprentissage pour la conduite de système*, 2006.

[Pomerol, 1995] J.-C. POMEROL. Artificial intelligence and human decision making. In *EURO XIV*, July 1995.

[Quesnel *et al.*, 2007] G. QUESNEL, R. DUBOZ, E. RAMAT, and M.K. TRAORÉ. Vle : A multimodeling and simulation environment. In *the Summer Simulation Multiconference*, pages pp. 367–374. (SummerSim'07), San Diego, California, USA, July 2007.

[Sarjoughian *et al.*, 2005] SARJOUGHIAN, Hessam HUANG, and DONGPING. A multi-formalism modeling composability framework : Agent and discrete-event models. In *DS-RT '05 : Proceedings of the Ninth IEEE International Symposium on Distributed Simulation and Real-Time Applications*, pages 249–256, Washington, DC, USA, 2005. IEEE Computer Society.

[Schneider, 1996] D. K. SCHNEIDER. *Modélisation de la démarche du décideur politique dans la perspective de l'intelligence artificielle*. PhD thesis, Faculté des sciences économiques et sociale de l'Université de Genève, 1996.

[Searle, 1980] J. SEARLE. Minds, brains, and programs. In *The Behavioral and Brain Sciences*, pages 3 :417–424, 1980.

[Sebliotte, 1974] M. SEBLIOTTE. Agronomie et agriculture. essai d'analyse des tâches de l'agronome. *Cahiers de L'ORSTOM*, 1974.

[Sebliotte, 1990] M. SEBLIOTTE. Système de culture, un concept opératoire pour les agronomes. *Un Point Sur... L. Combe ; D. Picard, inra edition*, 1990.

[Tortrat, 2005] F. TORTRAT. *Modélisation orientée décision des processus de transfert par ruissellement et subsurface des herbicides dans les bassins versants agricoles*. PhD thesis, ENSA de Rennes and INRA-Agrocampus Rennes UMR Sol Agronomie Spatialisation Rennes-Quimper, 2005.

[Trépos, 2008] R. TRÉPOS. *Apprentissage symbolique à partir de données issues de la simulation pour l'aide à la décision. Gestion d'un bassin versant pour une meilleure qualité de l'eau*. PhD thesis, Université de Rennes 1, 2008.

[Uhrmacher and Arnold, 1994] A. M. UHRMACHER and R. ARNOLD. Distributing and maintaining knowledge : Agents in variable structure environment. 1994.

[Van der Aalst and ter Hofstede, 2005] W. M. P. Van der AALST and A. H. M. ter HOFSTEDE. Yawl : Yet another workflow language. *Information Systems*, 30(4) :245–275, June 2005.

[Wooldridge, 2002] M. WOOLDRIDGE. *An Introduction to MultiAgent Systems*. Feb. 2002.

[Wooldridge and Jennings, 1995] M. WOOLDRIDGE and N. R. JENNINGS. *Intelligent agents : Theory and practice*. Knowledge Engineering Review, 1995.

[Zeigler, 1976] B. P. ZEIGLER. *Theory of Modeling and Simulation*. 1976.

[Zeigler *et al.*, 2000] Bernard P. ZEIGLER, Tag Gon KIM, and Herbert PRAEHOFER. *Theory of Modeling and Simulation*. Orlando, FL, USA, 2000.

Institut National de la Recherche Agronomique

Unité BIA, INRA Toulouse

Weisenhorn Geoffroy

Résumé :

Comment modéliser et simuler la Décision ?
Ce stage s'est déroulé au sein de l'INRA dans l'unité BIA à Toulouse.

Le but du stage a été de comprendre et de développer un modèle de décision de systèmes de culture pouvant être modélisé et simulé en DEVS, grâce à la plateforme RECORD-VLE, en utilisant des formalismes de la décision existant.

Mots-clefs :

Décision, Planification, Réseau de Petri, Modélisation, Simulation, DEVS, événements discrets, spécification, itinéraire techniques, raisonnements sur les actions, Modéle de Markov, Système Multi-Agents